A representation of cuts within 6/5 times the edge connectivity with applications

András A. Benczúr*
Department of Mathematics, M. I. T.

June 28, 1999

Abstract

Let G be an undirected c-edge connected graph. In this paper we give an $O(n^2)$-sized planar geometric representation for all edge cuts with value less than $\frac{6}{5}c$. The representation can be very efficiently built, by using a single run of the Karger–Stein algorithm for finding near-mincuts. We demonstrate that the representation provides an efficient query structure for near-mincuts, as well as a new proof technique through geometric arguments. We show that in algorithms based on *edge splitting*, computing our representation $O(\log n)$ times substitute for one, or sometimes even $\Omega(n)$, u–v min-cut computations; this can lead to significant savings, since our representation can be computed $\tilde{\Theta}(m/n)$ times faster than the currently best known u–v min-cut algorithm.

Key words: approximate minimum cuts, edge connectivity, edge splitting, undirected graphs, graph algorithms.

*E-mail: `benczur@theory.lcs.mit.edu`. Address: Laboratory for Computer Science, Theory Group, 545 Technology Square, Cambridge MA 02139. Research supported by DARPA contract N00014-92-J-1799 and NSF contract 9302476-CCR.

1 Introduction

In the paper we describe a data structure that handles all *approximately* minimum cuts: we represent them by a simple geometric structure consisting of polygons with some of their diagonals drawn in the plane. Our approach is entirely new in relating edge connectivity to geometry. In the discussion, we consider *undirected* graphs G with non-negative edge weights over the vertex set V. A *cut* is a partition of V into two non-empty subsets; the *value* of the cut is the total weight of edges connecting the two subsets. The *connectivity* c of the graph is the minimum value of its cuts. A *mincut* is a cut with value c; an δ*-near-mincut* is a cut with value less than δc.

We generalize the *cactus representation* [4] of all minimum cuts. We improve and extend the algorithmic applications of minimum cuts and especially the cactus (as e.g. in [22, 1, 15]). An implication of our theorem is that the number of cuts with value less than $6/5$ times the minimum cut value is at most $\binom{n}{2}$. Two recent results [21, 25] extend our results in bounding the *number* of approximate min-cuts: they prove for improved constants $4/3$ and $3/2$ instead of our $6/5$. However these results do not yield compact structural descriptions. So far, the only known improvement of the cactus representation on the structural side is the representation of minimum and minimum+1 cuts by Dinitz [5].

We present a randomized $\tilde{O}(n^{2(1+\delta)})$-time[1] algorithm to build our representation of δ-near-mincuts. The algorithm is based on the *recursive contraction algorithm* of Karger and Stein. We build our representation without exceeding the asymptotic running time of the Karger–Stein algorithm. We demonstrate that our representation serves as a query structure to the output of that algorithm.

Our representation extends for near-minimum cuts within a vertex subset W as well. For all cuts that separate W and have value less than $6/5$ times the minimum value over all such cuts, we may represent the intersection of cut sides with W. Unfortunately, this representation can only be built by using the fairly complicated and inefficient edge-splitting algorithm of Mader [18, 10]. Also note that although the number of such cuts is potentially exponential, most of them are identical if considered only within W: by our result, there are at most $\binom{|W|}{2}$ different possibilities for them.

Our main reason for considering near-mincuts is to speed algorithms up in the following type of applications. Assume the min-cuts have to be repeatedly found in a dynamically changing graph. Then by computing near-mincuts instead of min-cuts in a step, we may not have to recompute them in subsequent steps. Our main application is an algorithm that *splits off* edges from a prescribed vertex without decreasing the edge connectivity [17, problem 6.53], [18, 7, 10]. Another application of near-mincuts to the *edge augmentation* problem [27, 8] is described in [1].

1.1 Overview

The paper is organized as follows. In the rest of the Introduction, we review three topics—the edge splitting problem [17, 18], the efficient randomized (near-)mincut algorithm of Karger and Stein [15], and the cactus representation of min-cuts [4]—that serve as a background for the rest of the paper. The definition and the main properties of the polygon representation is described in Section 2. In Section 3, we give the high-level proof ideas. First we describe the representation of near-minimum cuts within a vertex subset W. The proof is then by inductively increasing the set W. In the same section, a new proof of the cactus representation along these lines is also given.

The technical details are found in Sections 4–6. Section 4 contains the existence proof. Based on this proof and the Karger–Stein algorithm [15], an efficient algorithm is described in Section 5.

[1] We use the $\tilde{O}$ (soft-O) notation to give asymptotics up to a polylogarithmic factor.

Our main application, a new proof and algorithm to perform edge splitting, is in Section 6.

1.2 Edge splitting

The main application of our near-mincut representation, both algorithmically and as a proof technique, is related to edge splitting. Let G be graph G with a distinguished vertex s.

Definition 1.1 (Edge splitting). For a pair of vertices $u, v \in G$ adjacent to s, we say that we *split us and vs by weight w* if we decrease the weights of these edges by w and increase that of uv by w. We say that we *split s off* if, by a sequence of splits, we isolate s from the graph.

Edge splitting is a tool widely used to solve connectivity-related problems. If the graph arising by splitting-off preserve the connectivity properties of the original graph, we may use this construction in inductive proofs and recursive algorithms for various problems. For a survey of applications as a proof technique see [9], and for efficient algorithms see [10]. The base of all these applications is the following two theorems (on the necessity and extension of the technical conditions in these theorems, see [7]):

Theorem 1.2. **(Lovász [17, problem 6.53])** *Let G be a weighted undirected graph with a distinguished vertex s. Let c' be the minimum value among the cuts of G other than the cut separating s from $V - s$. Then it is possible to split s off, such that the resulting graph has connectivity c'. Furthermore if all edge weights are integers, $c' \geq 2$, and the total weight of edges incident to s is even, then it is possible to give a sequence of splits by integer weights.*

Theorem 1.3 (Mader [18]). *Let G and s be as in the previous theorem. Then it is possible to split s off such that for all pairs of vertices $s, t \in V - s$, the minimum value of a cut separating s and t remains the same. A sequence of splits by integer weights can be found under the same conditions as in Theorem 1.2.*

In the paper, we concentrate on the algorithmic aspects of Lovász' Theorem. The previous best known algorithm for constructing the required splitting runs in $\tilde{O}(n^2 m)$ time [10]. We present a randomized algorithm for the same task that runs in $\tilde{O}(n^2 \max\{n, d(s)/c'\})$ time (here $d(s)$ is the total weight of edges incident to s). In certain applications of Lovász' Theorem, we may take advantage of the fact that $d(s) \approx c'$. Such an example is the *graph orientation* problem, where one has to find a k-connected orientation of an undirected $2k$-connected graph [17, problem 6.54]. We solve this problem in randomized $\tilde{O}(n^3)$ time. The best known deterministic time is $\tilde{O}(nm^2)$ [10].

We improve on previous splitting-off algorithms in the following way. First we show how to find splittable pairs that preserve the connectivity, given our representation. This step is based on the strong geometric properties of our representation. Notice that although one can find a single split by the knowledge of all minimum and minimum+1 cuts, it is unclear how to find such a split efficiently by the representation of Dinitz [5].

Our second improvement step is another version of the splitting-off theorems, stating that splitting-off is possible, without decreasing the value of any 6/5-*near-mincut*. This result is stronger than Lovász' and is independent of Mader's Theorem, and of theoretic interest as well. Based on the polygon representation, we can find splittable pairs in this stronger sense. Since all 6/5-near-mincuts are preserved, we may find splits with a total weight $\Theta(c')$ of edges that preserve all min-cuts (or all δ-near-mincuts for $\delta < 6/5$).

1.3 The algorithm and a query structure for Karger–Stein

A reason why near-mincuts can be algorithmically useful is that finding even *all* of them seems to be easier than finding the "classical" u–v min-cuts. The randomized algorithm of Karger and Stein [15] for finding near-mincuts "...breaks the $\Omega(nm)$ max-flow barrier for finding min-cuts ...[and] suggests that the [global] min-cut problem might be fundamentally easier to solve..." [15]. The time to find δ-near-mincuts by this algorithm is $\tilde{O}(n^{2(1+\delta)})$. While finding near-mincuts is in RNC [13], Mulmuley's results [19] indicate that this is unlikely for u–v min-cuts, provided edge weights are sufficiently large.

One may ask why our representation is necessary to achieve improved running times, when the Karger–Stein algorithm already "finds" all δ-near-mincuts. Notice however that there can be as many as $\Omega(n^2)$ min-cuts of a graph, requiring $\Omega(n^3)$ space to list all of them. This amount of space exceeds the running time $\tilde{O}(n^{2(1+\delta)})$ of the Karger–Stein algorithm; consequently, that algorithm finds an (implicit) representation of the cuts.

In conclusion, there is a hope that algorithms can be improved for problems where the knowledge of (almost) all near-mincuts is needed, and our representation contains crucial information about cuts that is not straightforward to "read" from the output of the Karger–Stein algorithm. In such a case, our representation can then serve as a *query structure* to near-mincuts.

For the case of min-cuts, the *edge augmentation problem* [27, 22] illustrates the phenomenon when the required knowledge about min-cuts is the cactus representation. In our main application, the *edge splitting* problem, the situation is similar. Although the knowledge of all near-mincuts determines a solution, it is still not straightforward to find one asymptotically within the running time of the Karger–Stein algorithm. Here the geometric properties of our representation are extensively used.

We give an algorithm for transforming the output of the Karger–Stein algorithm to our near-mincut representation, without exceeding the asymptotic running time of the Karger–Stein algorithm. For the case of the cactus of min-cuts, such algorithms were given independently in [15] and [1]. These algorithms both follow the steps of (two slightly different) cactus-proofs. In Section 5, we follow the same approach for near-mincuts, using our proof from Section 4. We remark that the algorithms of [21] or [26] for enumerating all near-mincuts give rise to *deterministic*, though not very efficient, algorithms to find our representation.

1.4 The cactus representation

In this section, we describe the *cactus representation* of all min-cuts. We give a description that is slightly modified compared to the usual discussion in the literature, in order to motivate the definition of our representation.

A *cactus* is a graph which contains no cut edges and no two cycles with a common edge. In other words, a cactus is built up from a single vertex by recursively joining cycles (of length possibly two!) to existing vertices. The *cactus representation* (Fig. 1) of a non-disconnected graph G is a cactus $\mathcal{K}$ such that a partition of V corresponds to the vertex set of the cactus. The min-cuts of the cactus are precisely those which arise by erasing two edges of a cycle. Then the min-cuts of G are precisely the edge sets of G connecting two components of a min-cut of the cactus.

Next we reformulate the description of the cactus representation. The *cactus representation* consists of a collection of cycles. To each cycle, there is a partition of the vertex set into *atoms*, such that atoms and vertices of the cycle are mapped to each other. By cutting one such cycle into two, the union of the atoms on the two sides of the cycle form sides of a cut of the graph. The collection of min-cuts is equal to the collection of such cuts defined by the cycles.

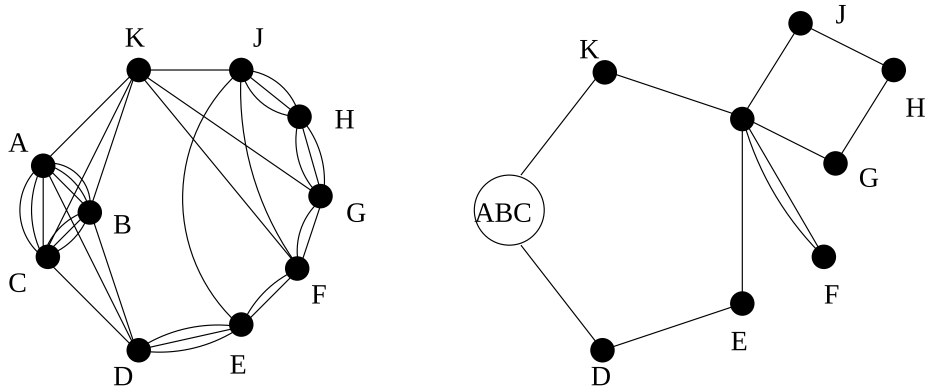

Figure 1: A graph with the *cactus representation* of its min-cuts of value 6. The cactus consists of three cycles, with one vertex from each cycle contracted. The *connecting atoms* corresponding to the contracted vertices are FGHJ, A...FK and A...EG...K. The union of any pair of these three sets is the entire vertex set.

In the original definition of the cactus representation, an atom corresponding to a vertex v in a specific cycle can be found by taking the union of all vertices reachable from v without using edges from the cycle. The following definition captures the properties of the atoms, given that the cycles are arranged in a tree-like fashion.

Definition 1.4 (Tree-hierarchy). Let $\mathcal{P}_1, \ldots, P_\ell$ be a collection of partitions of the vertex set V into atoms. For each i, let G_i be graph with $\mathcal{P}_i$ as vertex set and an arbitrary cycle as edge set. Assume that a cactus graph $\mathcal{K}$ is given, with cycles G_i. We say that $\mathcal{P}_1, \ldots, P_\ell$ possess a *tree-hierarchy* if for all vertices w of $\mathcal{K}$ and any two cycles G_i and G_j containing w, the atoms $Z_i \in \mathcal{P}_i$ and $Z_j \in \mathcal{P}_j$ corresponding to w satisfy $Z_i \cup Z_j = V$ (see Fig. 1). We say that Z_i and Z_j are the *connecting atoms* of $\mathcal{P}_i$ and $\mathcal{P}_j$.

Note that the choice of the cactus is arbitrary in the definition. We could have replaced cycles by complete graphs, or paths, for example. A nice data structure to handle the tree-hierarchy is a P–Q tree as noticed in [16].

Theorem 1.5 (Dinitz, Karzanov, Lomonosov [4]). *All min-cuts of an undirected graph G can be represented by a cactus representation. The representation consists of a collection of cycles, such that the min-cuts of G are precisely those represented by the cycles. The partitions corresponding to cycles possess a tree-hierarchy.* □

2 The main result

In this section, we describe the representation for all 6/5-near-mincuts. This main result will be proved by the end of Section 4. An efficient algorithm to build the representation, based on the proof, is given in Section 5. The representation consists of convex *representing polygons*, generalizing the cycles of the cactus.

Definition 2.1 (Polygon representation). *The polygon representation of a collection of cuts $\mathcal{C}$ consists of a collection of representing polygons over a partition of the vertex set; these partitions possess a tree-hierarchy as defined in Definition 1.4.*

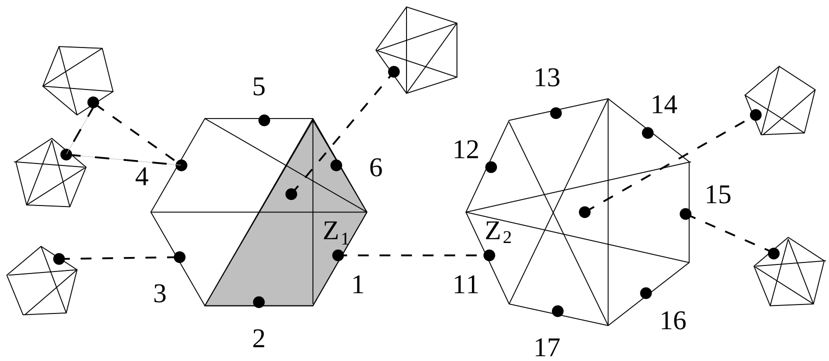

Figure 2: A collection of *representing polygons*. Dots denote the positions of the *atoms*. The shaded region is a *diagonal side*, representing the union of the four atoms in that region. Dashed edges are between *connecting atoms* (Definition 1.4). For two of the polygons, the *outside atoms* are 1 through 6 and 11 through 17, in cyclic order (Section 2.2).

- *A representing polygon is a polygon with a collection of distinguished representing diagonals, with all polygon-edges and diagonals drawn by straight lines in the plane (see Fig. 2).*
- *The representing diagonals divide polygons into cells.*
- *For a representing polygon, there is a corresponding partition of the vertex set into atoms.*
- *Each atom corresponding to a polygon is mapped to a cell. However, some cells may contain no atoms (typically, most of them are empty).*
- *Each representing diagonal defines a cut, with sides as the union of atoms contained by cells on the diagonal sides.*
- *The collection of cuts $\mathcal{C}$ equals to the collection of cuts defined by representing diagonals of all polygons.*

Notice that a cycle of the cactus representation can be transformed in the straightforward way (see Fig 3) to a polygon with the same number of polygon-edges and *all* diagonals drawn. However, while each single atom Z of the cactus is the side of a min-cut, these cuts may not be included in a single polygon as in the above definition. In general, we have to be concerned about representing single cuts: we allow "polygons" with two edges and a single diagonal, to represent a single cut analogously to a length-2 cycle of the cactus representation. We address the issue of such degenerate polygons to more extent in Section 2.3.

We can now bound the size of the polygon representation and derive an $\binom{n}{2}$ bound on the number of 6/5-near-mincuts. This bound matches the well-known bound for the number of min-cuts (as e.g. implied by the cactus representation [4]) and the more recent bound for 4/3-near-mincuts by Nagamochi et al. [21]. In order to bound the size of the representing polygons, we have to impose two additional requirements:

- there must be at least one diagonal incident to every vertex of a polygon; and
- there must be some atom mapped to each cell adjacent to a polygon-edge.

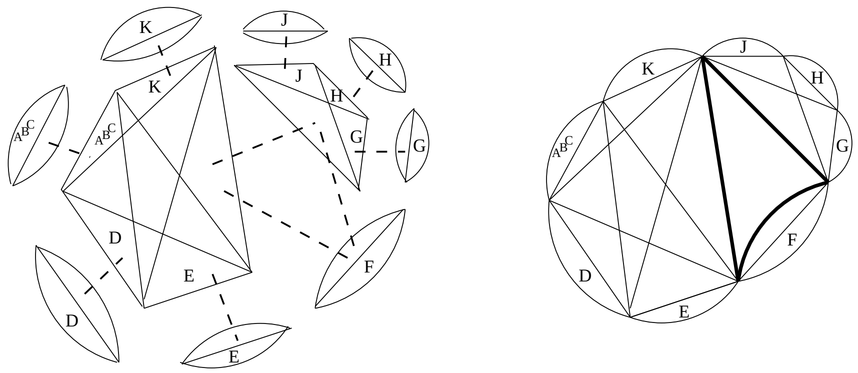

Figure 3: *Left:* The cycles of the cactus representation of Fig. 1 turned to representing polygons. Notice that extra degenerate polygons are added to represent single cuts. *Right:* The cactus representation can be joined to a single polygon.

Theorem 2.2. *In a polygon representation, both the total number of atoms and the total number of polygon-vertices is at most $O(n)$. Hence if the 6/5-near-mincuts can be represented by diagonals of a hierarchical collection of polygons, then the number of 6/5-near-mincuts is at most $\binom{n}{2}$.*

Proof. Let us consider the cactus $\mathcal{K}$ defining the hierarchy-tree of the polygon-representation. Each representing polygon corresponds to a cycle in this cactus. The number of atoms in a polygon equals the length of the corresponding cycle, which in turn is an upper bound on the number of edges. The first part follows by the size bound on the cactus representation [4], see also [23]. Finally, the possible number of cuts in a polygon representation is upper bounded by the number of possible cuts of the cactus representation $\mathcal{K}$, proving the second part. □

2.1 Configurations of non-minimum cuts: cycles and inside atoms

We saw that the cycle of a cactus representation can easily be turned to a representing polygon. A special property of the polygons arising in this way is that all atoms are mapped to cells adjacent to the boundary of the polygon. The main reason why polygons are more powerful is that atoms may get mapped to other "inside" cells as well. An example for a graph in which the near-mincut representation has atoms in the inside is a *wheel* with sufficiently large weights on the outside cycle, see Fig 4. The specific configuration of cuts in this example is abstracted as follows:

Definition 2.3. We say that k cuts $(C_i|\overline{C}_i)$ for $i \leq k$ form a k-*cycle* if

- $(C_i|\overline{C}_i)$ crosses both $(C_{i-1}|\overline{C}_{i-1})$ and $(C_{i+1}|\overline{C}_{i+1})$;
- $C_i \cap C_j = \emptyset$ if $i \neq i-1, i$ or $i+1$; and
- $Z = \bigcap_{i \leq k} \overline{C}_i \neq \emptyset$.

Finally, we classify cells (and atoms) depending on their adjacency with the polygon-boundary (see Fig. 5). We call a set of cells given by the intersection of some diagonal sides a *region*. We say that a region is

- *inside* if it is not adjacent to the boundary of the polygon;
- *outside* if it has a contiguous segment of adjacent boundary edges;

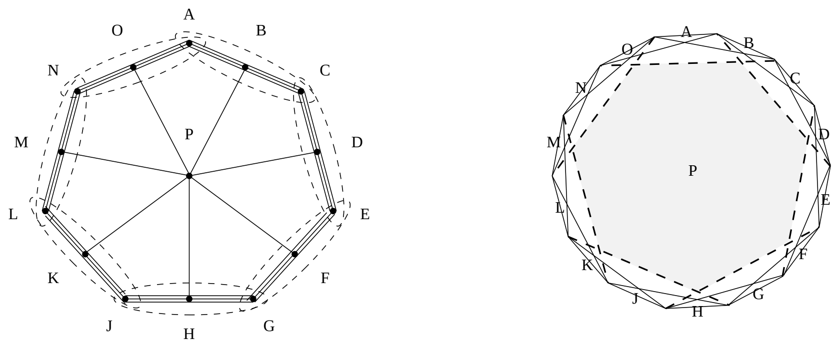

Figure 4: An example of a *k-cycle* formed by cuts of value 7 in a graph with connectivity 6. In the polygon representation (*right*), P is an *inside atom*.

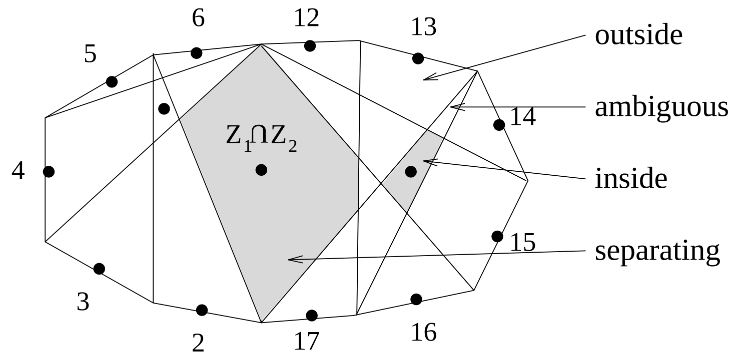

Figure 5: A single polygon, with examples for types of cells. The polygon arises by joining the two large polygons of Fig. 2 (Section 2.3).

- *separating* if it is adjacent to a non-contiguous part of the polygon boundary; and
- *ambiguous* if it is adjacent to a single polygon-vertex.

2.2 Beyond geometry

In general, the arrangement of cells is dependent on the geometric position of the vertices of the polygon. By giving different coordinates, some regions may vanish, or new cells may appear. Thus it is not necessarily true that the positions of all atoms can be found, by arbitrarily positioning the polygon-vertices. Can we assume, for example, that the representing polygons are regular, or that all of their vertices have integer coordinates? Fixing such an actual geometric position of the polygons will be important both in the proof and in algorithmically handling the representation.

The main result in this section implies that the actual geometric positions of polygon-vertices is irrelevant. Interestingly, this result is an implication of Helly's theorem in 2 dimensions [6]. This fact implies another compact representation of all near-mincuts: a cyclically ordered system of pairwise disjoint subsets of V (corresponding to outside atoms, Fig 2), where each near-mincut (or diagonal) side can be uniquely described by the clockwise starting and ending subsets contained by it.

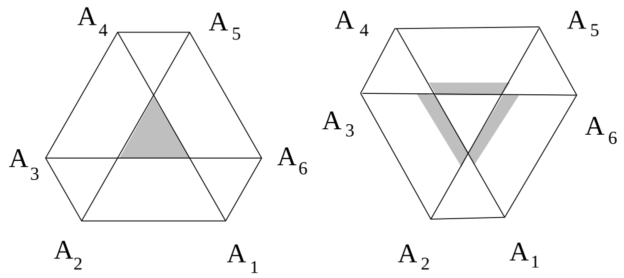

Figure 6: A cell Z contained by the shaded region vanishes by deforming the polygon.

In contrast to our goal, consider an inside cell Z surrounded by three diagonals forming a 3-cycle as in Definition 2.3 (A_1A_4, A_2A_5, and A_3A_6 in Fig. 6). Then by an appropriate deformation of the polygon, the three diagonal sides that contained Z no longer have a common intersection. Our main theorem is that the only reason why a representing polygon could not be arbitrarily deformable is the existence of 3-cycles. Later in Section 2.4, we will prove that for 6/5 (and in fact, 4/3) near-mincuts, such a configuration is impossible. For the purposes of this section, we add this fact as an extra assumption. Except for this assumption (iii) below, all conditions of the theorem are satisfied if $\mathcal{C}$ is a collection of cuts represented by a polygon with ℓ edges and with atoms $X_1, \ldots, X_\ell$ adjacent to them.

Theorem 2.4. *Let $\mathcal{C}$ be a collection of cuts of G. Let a collection $X_1, \ldots, X_\ell$ of pairwise disjoint cyclically ordered vertex subsets be given; let $X = \bigcup X_i$. Assume that*

(i) each cut side in $\mathcal{C}$ intersects X in a consecutive subset of the X_i's;

(ii) if two cut sides C_1 and C_2 of $\mathcal{C}$ have $C_1 \cap X$ and $C_2 \cap X$ disjoint, then they are disjoint as well; and

(iii) $\mathcal{C}$ contains no three cuts forming a three-cycle.

Then an arbitrary convex ℓ-gon $\mathcal{P}$ can be extended to a polygon representation of $\mathcal{C}$. For each atom, the corresponding cell is uniquely determined.

Proof. We assign the X_i to the edges of $\mathcal{P}$ in a cyclic order. We draw a diagonal with one side containing $X' = X_1 \cup X_{i+1} \cup \ldots \cup X_j$ if there is a cut of $\mathcal{C}$ with side C such that $C \cap X = X'$. We prove that all remaining vertices can be assigned to cells of this polygon; hence this polygon represents $\mathcal{C}$.

Let us consider a vertex $s \in V - X$. Let $\mathcal{S}$ be the collection of diagonal sides corresponding to cut sides of $\mathcal{C}$ containing s. We show that the intersection of $\mathcal{S}$ is non-empty. Each diagonal side is a convex subset of the plane. By Helly's theorem (see e.g. [6]), if a collection of convex subsets of the plane has the property that all three of them has a non-empty intersection, then all of them have a common intersection. Next we prove that the requirement of this theorem holds for $\mathcal{S}$. Then the non-empty intersection of $\mathcal{S}$ must be a unique cell, since each diagonal may contain s on only one of its two sides.

Let us consider three diagonal sides C_1, C_2 and $C_3 \in \mathcal{S}$. We are done if $C_1 \cap C_2 \cap C_3 \cap X$ is non-empty: then the three diagonal sides have a common intersection adjacent to the polygon-boundary. By (ii), each pair $i < j \leq 3$ must have $C_i \cap C_j \cap X$ non-empty, since $s \in C_i \cap C_j$. We prove that the only possible case when the three diagonal sides have a non-empty intersection is when they form a 3-cycle, contradicting (iii). We know that $C_1 \cap C_2$, $C_1 \cap C_3$ and $C_2 \cap C_3$ are pairwise disjoint non-empty sets (intersecting X). But then these sets and s prove that the C_i are pairwise crossing. Finally, $s \in \bigcap C_i$, hence Definition 2.3 is satisfied. □

2.3 Subdivision of cuts into polygons

We finish the description of the polygon representation by describing how and why does the system of represented cuts get divided into distinct representing polygons. This entire section is based on Theorem 2.4 and the "ambiguous" role of separating and ambiguous atoms. Note that an ambiguous or separating atom could be included in the system of X_i in Theorem 2.4; in fact, a separating atom could have a multiple occurrence as well. We start by stating the main result (see Fig. 5):

Theorem 2.5. *If a polygon contains a separating cell, it can be subdivided into two polygons. The cycles coresponding to these polygons are incident in the tree-hierarchy, with outside or separating connecting atoms. And if two polygons with more than two edges (non-degenerate polygons) are incident in the tree-hierarchy with non-inside connecting atoms, then they may be joined to a single polygon. This polygon contains a separating cell.*

Proof. To prove the first part, let us build the sequence of outside atoms $X_1, \ldots, X_\ell$ between two polygon-vertices adjacent to the separating cell. Let $Y_1, \ldots, Y_k$ be the sequence of the remaining outside atoms. Let $X = \overline{\bigcup X_i}$; $Y = \overline{\bigcup Y_i}$. Now the two sequences $X, X_1, X_2, \ldots$ and $Y, Y_1, Y_2, \ldots$ define the two polygons. The connecting atoms are X and Y; they satisfy $X \cup Y = V$.

For the second part, we do the opposite procedure with two sequences $X, X_1, X_2, \ldots$ and $Y, Y_1, Y_2, \ldots$ of outside atoms with $X \cup Y = V$. Then we form the new sequence $X_1, \ldots, X_\ell, Y_1, \ldots, Y_k$. In this polygon, there is a separating cell with adjacent polygon-vertices between X_1, Y_k and X_ℓ, Y_1. If $X \cap Y$ is non-empty, it is an atom placed to this separating cell. □

It will be crucial in our proof (Section 4) that all possible polygons are joined and there are no pairs of non-inside connecting atoms. Unfortunately, degenerate polygons are exceptions in Theorem 2.5. Let there be two adjacent 2-edge degenerate polygons in the hierarchy. Their connecting atoms are clearly inside; however if we join them, we get a 2-edge polygon with two distinct "diagonals" connecting the two edges. Such a polygon (even if we allow diagonals connecting incident vertices) is an exception of (i) in Theorem 2.5. One way to get around is to allow this exception; another way is to add an extra edge between the two diagonals.

Corollary 2.6. *In the coarsest subdivision of cuts into polygons, there are no pairs of non-inside connecting atoms. In the finest subdivision, there are no separating cells.* □

Finally, the theorem below will serve as an assumption on the polygon representation. The proof is trivial since an ambiguous atom can be added to the X_i-sequence in Theorem 2.4.

Theorem 2.7. *We may assume that none of the representing polygons contains ambiguous atoms.* □

2.4 Excluded configurations and conclusions

So far in Section 2, we have defined the polygon representation of a collection of cuts. We have introduced the notion of *cycles of cuts*, as a configuration characteristic to the representation. We have shown that if a collection of cuts contains no 3-cycles and can be represented by (convex) polygons, then these polygons can arbitrarily be deformed and the representation can still be recovered.

Apart from 3-cycles, we describe another important excluded configuration such that a system of cuts containing it cannot be represented by diagonals of polygons. The configuration is (basically) the well known *comb* with three teeth [24, 3, 16]. The inexistence of combs gives the tightness of the 6/5-bound in our theorem.

Definition 2.8. Four cuts $(C_1|\overline{C}_1)$, $(C_2|\overline{C}_2)$, $(C_3|\overline{C}_3)$ and $(D|\overline{D})$ form a *comb* if $C_i \cap D \neq \emptyset$ for $i = 1, 2, 3$ and $C_1 - D$, $C_2 - D$ and $C_3 - D$ are pairwise disjoint.

Our main theorem can be summarized as follows (the proof is in Section 4):

Theorem 2.9 (Main result). *Let δ be such that the collection of δ-near-mincuts contains no combs, no 3-cycles and no 4-cycles. In particular, this is the case if $\delta = 6/5$. Then δ-near-mincuts possess a polygon representation, in which the representing polygons can be arbitrarily deformed.*

By the following sequence of lemmas, we prove that the system of 6/5-near-mincuts indeed have the required properties. We summarize some definitions we use in our proofs. In an edge weighted graph $G = (V, E)$, let $d(X, Y)$ denote the total weight of edges connecting $X, Y \subset V$. If it causes no ambiguity, we denote $V - C$ by $\overline{C}$. We denote a cut with sides C and $\overline{C}$ by $(C|\overline{C})$. We say that C is a side of a (near-)mincut if $(C|\overline{C})$ is a (near-)mincut. We let $d(C) = d(C, \overline{C})$; $d(C)$ is the value of the cut $(C|\overline{C})$. We say that two sets C and D are *crossing* all of their four components are non-empty. Note that if C and D are represented by diagonal sides, then C and D are crossing precisely if the corresponding diagonals cross.

The first lemma below is the generalization of the following well-known property of min-cuts:

Fact 2.10. *Let $(C|\overline{C})$ and $(D|\overline{D})$ be two crossing min-cuts. Then $d(C \cap D, C - D) = c/2$; consequently, $C \cap D$ and $C \cup D$ are both min-cut sides as well.* □

Lemma 2.11 (Perimetric concentration). *Let $(C|\overline{C})$ and $(D|\overline{D})$ be two crossing $(1+\epsilon)$-near-mincuts. Then $d(C \cap D, \overline{C} \cap D) \geq (1 - \epsilon)c/2$.*

Proof. We have $d(\overline{C} \cap D) \geq c \geq d(D) - c\epsilon$. Since edges connecting $\overline{C} \cap D$ to $\overline{D}$ are counted on both sides, by subtracting them we get that $d(C \cap D, \overline{C} \cap D) + c\epsilon \geq d(C \cap D, \overline{D})$. Since $c \leq d(C \cap D) = d(C \cap D, \overline{C} \cap D) + d(C \cap D, \overline{D})$, combining the last two inequalities gives $c \leq 2d(C \cap D, \overline{C} \cap D) + c\epsilon$, which proves the result. □

Lemma 2.12 (Excluded Combs). *There are no three 6/5-near-mincuts D_1, D_2 and D_3 which cross another C with $D_i \cap C$ pairwise disjoint.*

Proof. The edge sets connecting $D_i \cap C$ and $\overline{C}$ are disjoint, for $i = 1, 2, 3$. By the Perimetric Lemma, each of them has weight at least $(1 - 1/5)c/2 = 2/5 \cdot c$. But then $d(C) \geq 6/5c$, contradicting that C is near-minimum. □

Lemma 2.13 (Excluded Cycles). *As in Definition 2.3, let $(C_i|\overline{C}_i)$ for $i \leq k$ form a k-cycle with $Z = \bigcap_{i \leq k} \overline{C}_i$. Then $\delta > 1 + 1/k$. In particular, 6/5-near-mincuts may not contain k-cycles for $k \leq 5$.*

Proof. By counting the number of occurrences of the edges on both sides, we get that

$$d(C_i) \geq d(C_i \cap C_{i+1}) + d(C_i \cap C_{i+1}, Z) + d(C_i - (C_{i-1} \cup C_{i+1}), Z) \ ,$$

$$d(Z) = \sum_{i \leq k} \Big(d(C_i \cap C_{i+1}, Z) + d(C_i - (C_{i-1} \cup C_{i+1}), Z) \Big) \ .$$

We also have that $C_i \cap C_{i+1} \neq \emptyset$, whence $d(C_i \cap C_{i+1}) \geq c$. By summing the first inequality for $i \leq k$, we obtain

$$\sum_{i=1}^{k} d(C_i) \geq d(Z) + \sum_{i=1}^{k} d(C_i \cap C_{i+1}) \ .$$

If Z is non-empty, the right hand side is at least $(k+1)c$, while the left hand side is less than $kc\delta$. Hence $\delta > 1 + 1/k$. □

By the previous lemma, we completed our goal. As a last remark, we reformulate of the inexistence of 3-cycles to a convenient form. This corollary is a well-known fact, see [17, 12].

Corollary 2.14. (Three-sets submodularity, [17, problem 6.48]) *If $(C_1|\overline{C}_1)$, $(C_2|\overline{C}_2)$ and $(C_3|\overline{C}_3)$ are 4/3-near-mincuts, then at least one of the four sets*

$$C_1 \cap C_2 - C_3, \quad C_1 \cap C_3 - C_2, \quad C_2 \cap C_3 - C_1, \quad \overline{C}_1 \cap \overline{C}_2 \cap \overline{C}_3$$

must be empty. By exchanging the roles of C_i and $\overline{C}_i$, the claim also applies for

$$C_1 \cap C_2 \cap C_3, \quad \overline{C}_1 \cap \overline{C}_2 \cap C_3, \quad C_1 \cap \overline{C}_2 \cap \overline{C}_3, \quad \overline{C}_1 \cap C_2 \cap \overline{C}_3 \ .$$

Proof. If all four sets are non-empty, the three cuts for a three-cycle. □

3 Proof framework

The main goal of this section to prepare the inductive existence proof of the polygon representation. In the first subsection, we show that the representation can be restricted to an arbitrary vertex subset; in the proof we will then step-by-step increase this subset. The second subsection describes the main difficulty in this procedure: if we add an extra vertex s to the existing representation, (ii) of Theorem 2.4 may fail since s may have to get added to disjoint pairs of diagonal sides. Finally, Section 3.3 contains a new proof of the cactus representation, as an illustration of the main ideas for the case of min-cuts.

3.1 Representation of near-minimum cuts of a vertex subset

In this section we show how to represent cuts that separate a given subset W with value within $6/5$ times the local connectivity of W. This result is of its own interest; also, the proof and the algorithm to build the polygon-representation is based on inductively increasing the subset W and building representations within W. For $W \subseteq V$, we say that $(C|\overline{C})$ is a W*-cut* if both $C \cap W$ and $W - C$ are non-empty. The *(local) connectivity* of W is the minimum value of W-cuts. (For $W = \{u, v\}$, W-connectivity is the familiar notion of u–v connectivity.) For a subset W with connectivity c, a δ-near-minimum W-cut is a cut with value less than δc.

Definition 3.1. Let W be a vertex subset of G. The W*-representation* of collection of W-cuts $\mathcal{C}$ is the polygon representation of the system $\{(C \cap W | W - C) : (C|\overline{C}) \in \mathcal{C}\}$.

We remark it here that the constructions in this subsection are purely for proof purposes. First of all, given a W-representation, we may not know any of the near-minimum W-cuts—only their intersection with W. Hence for $W = \{s, t\}$, we can build the W-representation without even knowing the value of a minimum s–t cut. Second, the procedure in the proof below is algorithmically very inefficient because a splitting-off algorithm satisfying Mader's theorem is used as a subroutine.

On the positive side, we will see that the representation can be built, given a list of all cuts $\{(C \cap W | W - C) : (C|\overline{C}) \in \mathcal{C}\}$. When $|\overline{W}|$ is constant, it is a simple observation (see e.g. [1]) that near-minimum W-cuts can be found by the Karger–Stein algorithm. In our splitting-off algorithm (Section 6), we use the fact that a $W = (V - s)$-representation can be efficiently built for a fixed vertex $s \in V$.

Theorem 3.2. *If the δ-near-mincuts of any graph have a polygon representation, then the δ-near-minimum W-cuts of any graph and vertex subset W have a W-representation.*

Proof. Let us obtain a new graph G' by splitting off all vertices of G in $\overline{W}$. By Mader's Splitting Theorem (Theorem 1.3), we may perform splitting-off without decreasing the connectivity c of W. Since the connectivity of W remains unchanged and no cut value may increase, a 6/5-near-minimum W-cut of G corresponds to a 6/5-near-mincut of G'. Hence the polygon representation of G' contains all cuts $\{(C \cap W|W - C) : (C|\overline{C})$ is a 6/5-near-mincut of $G\}$.

Notice however that by splitting-off, cut values may *decrease* and a W-cut with initial value at least $6/5 \cdot c$ may become a 6/5-near-mincut of G'. We solve this problem by removing all diagonals corresponding to such "false" cuts. The structure of the polygon representation is such that the removal of diagonals is straightforward. □

3.2 Twin cuts: the inductive procedure to build the representation

We prove the existence of (and algorithmically build) our representation by recursion on the number of vertices: given a W-representation, we show how to build the $(W + s)$-representation, for some vertex $s \in \overline{W}$. The proof can be greatly simplified by considering the case $W = V - s$ only: the task for an arbitrary W and s may be reduced to the required form by splitting all vertices in $V - (W + s)$ off, using Mader's Splitting (Theorem 1.3). In fact, the splitting theorem is used only to derive the inexistence of 3, 4-cycles and combs for the system of 6/5-near-minimum W-cuts.

Throughout the proof, we let $W = V - s$. We consider a polygon $\mathcal{P}$ of the W-representation. For all cuts $(C|\overline{C})$ to be represented, there is an image $(C \cap W|W - C)$ in the W-representation. Assume that $s \in C$ for this cut; then there has to be a diagonal in the target representation corresponding to $(C \cap W|W - C)$; furthermore, this diagonal has to contain s on the $(C \cap W)$-side.

Definition 3.3. We say that a diagonal side of the W-representation containing $C \subset W$ *has to be extended by* s if $(C + s|W - C)$ is a 6/5-near-mincut. In a polygon $\mathcal{P}$ of the W-representation, we denote the set of such diagonal sides by $\mathcal{S}(\mathcal{P})$.

By Theorem 2.4, there is a cyclically ordered sequence of subsets $X_1, \ldots, X_\ell$ of W (the outside atoms) corresponding to polygon $\mathcal{P}$ that determine $\mathcal{P}$. If it happens to be the case that the system of cuts $\mathcal{C}$ of G whose image in W is represented by $\mathcal{P}$ satisfy (i–iii) of Theorem 2.4, then the augmented representing polygon is determined by the X_i.

The main reason why the above idea fails in general is the seemingly uninteresting condition (ii): diagonal sides that have to be extended by s may not be intersecting. A simple example is a pair of cuts $(C|\overline{C})$ and $(C - s|\overline{C} + s)$ with $s \in C$ to be represented; these two cuts map to the same diagonal of the W-representation and require that s belong to both sides simultaneously. In this case, the diagonal of the W-representation has to be doubled—in fact, the number of cuts we handle increase when we step from W to $W + s$. The key idea of the recent bound on the number of 4/3-near-mincuts by Nagamochi et al. [21] (extended to 3/2-near-mincuts by [25]) is to bound the amount of increase by the characterization of such pairs of cuts.

The main idea of our proof is the extension of the characterization of pairs of cuts $(C|\overline{C})$ and $(C - s|\overline{C} + s)$ as in [21]. The next definitions gives the key notion; the main characterization is then Theorem 3.7 of the next subsection. We remark it here that [2] outlines an alternate (algorithmically less efficient) proof without using the ideas of [21].

Definition 3.4. We say that two diagonal sides of the W-representation containing C_1 and $C_2 \subset W$ are *twins* if $C_1 \cap C_2 = \emptyset$ and both diagonal sides have to be extended by s. For a polygon $\mathcal{P}$, let

$\mathcal{S}'(\mathcal{P})$ denote those elements of $\mathcal{S}(\mathcal{P})$ that have no twin in $\mathcal{S}(\mathcal{P})$; let $\mathcal{S}''(\mathcal{P})$ denote those that have no twin at all.

For a polygon $\mathcal{P}$ of the W-representation, let $X_1, \ldots, X_k$ be the sequence of its outside atoms. The system of cuts of the *entire graph* mapped to $\mathcal{P}$ obviously satisfy (i) and (iii) of Theorem 2.4, with the sequence of the X_i. Finally, if a subsystem with no twin pair is to be represented only, then (ii) holds as well. This implies the following lemma:

Lemma 3.5. *Let $\mathcal{P}$ be a polygon of the W-representation, let $X_1, \ldots, X_k$ be the sequence of its outside atoms. Then in the entire graph, the sequence of the X_i satisfies Theorem 2.4 for any subset of the cuts represented by $\mathcal{P}$ that contain no twin pair in $\mathcal{P}$. In particular, if there are no twins in $\mathcal{P}$, then the X_i-sequence defines the representation in V. And in general, any subset $\mathcal{S}' \subseteq \mathcal{S}(\mathcal{P})$ containing no twin pair has $\bigcap \mathcal{S}'$ non-empty.* □

3.3 A new cactus proof

In this section, we show that the system of min-cuts can be represented by a single polygon. This result is no surprise: after transforming the cycles of the cactus to polygons, all polygons can be joined, as in Corollary 2.6. Hence what we prove is slightly weaker then the cactus, but the existence of the cactus representation could easily be derived. Our aim here is to illustrate the main proof technique on the simple example of min-cuts.

Theorem 3.6. *The system of min-cuts can be represented by a single polygon as described in Definition 2.1.*

Proof. The proof goes along the lines of the previous subsection: we assume a $W = (V - s)$-representation of min-cuts is given and show how to build the entire representation. First we give the main characterization of diagonal sides with twins. We prove a slightly more general theorem that we will use for 6/5-near-mincuts as well. Then in two further steps, we specialize this theorem to min-cuts. Note that the inexistence of 3, 4-cycles and combs for min-cuts follow from the lemmas in Section 2.4, but could very easily be derived from Fact 2.10.

Theorem 3.7. *Assume that for a system $\mathcal{C}$ of cuts that contain no twins, no 4-cycles, and satisfy the claim of Lemma 2.14, a $(V - s)$-representation by a polygon $\mathcal{P}$ is given. Let $\mathcal{P}$ contain diagonal sides $C_j \subset W$ with twin for $j = 1, 2, \ldots$; let the corresponding diagonals have end-vertices A_j and B_j. Then we may number the $A_j B_j$ such that $A_1, A_2, A_3, \ldots, B_3, B_2, B_1$ is a clockwise sequence of polygon vertices (i.e. the diagonals are "parallel").*

Proof. We prove that (i) there are no three pairwise disjoint diagonal sides with twin; and (ii) that no diagonal sides with twins may cross each other. These two facts imply the theorem. Assume first that (i) fails to hold. Then there are three pairwise disjoint diagonal sides C_1, C_2 and C_3 that have to be extended by s, i.e. $(C_1 + s|W - C_1)$, $(C_2 + s|W - C_2)$, $(C_3 + s|W - C_3)$ are all in $\mathcal{C}$. But these three cuts violate Lemma 2.14.

We prove (ii) by distinguishing three cases. Let D_1 and D_1' be twin pair, let D_2 and D_2' be another twin pair such that D_1 and D_2 cross. Assume first that $D_2' \cap D_1 = \emptyset$: then the components D_1, D_2, D_2' and s of the three near-mincuts $(D_1 + s|\overline{D}_1)$, $(D_2 + s|\overline{D}_2)$ and $(D_2' + s|\overline{D}_2')$ violate Lemma 2.14. Next if D_2' and D_1 are crossing, D_1, D_2 and D_2' form a comb. The only remaining case is when $D_1' \subseteq D_2 - D_1$ and similarly $D_2' \subseteq D_1 - D_2$. Then the four sets D_1, D_2, $D_1' + s$ and $D_2' + s$ form a 4-cycle. We reached a contradiciton in all three cases. □

Lemma 3.8. *Let C be a diagonal side that has to be extended by s. If C has a twin pair, then both $(C + s|W - C)$ and $(C|\overline{C})$ are min-cuts. Let C and D be a pair of twins; then $(C \cap D|\overline{C \cap D})$ is a min-cut.*

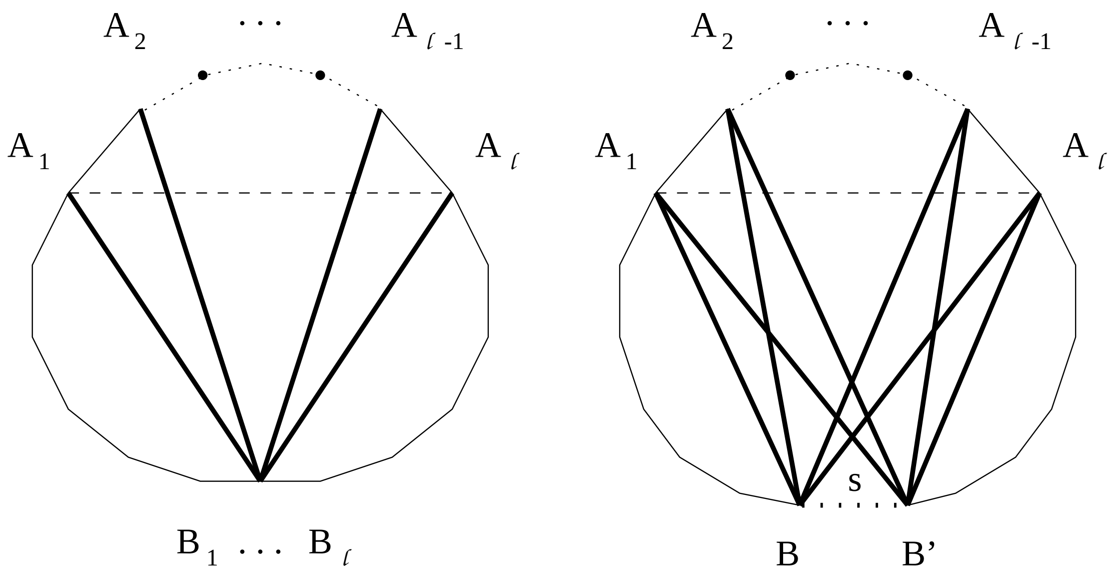

Figure 7: The inductive procedure to build the "cactus representation". Thick diagonals (A_iB_i) are as in Theorem 3.7.

Proof. By Fact 2.10. □

Corollary 3.9. *For a $W = (V - s)$-representation of all min-cuts, let there be ℓ diagonals A_iB_i, $i = 1, \ldots, \ell$, as in Theorem 3.7. Let A_1B_1 divide W into C_1 and $W - C_1$; similarly let C_ℓ and $W - C_\ell$ belong to $A_\ell B_\ell$, such that $C_1 \cap C_\ell = \emptyset$. Then $(C_1 \cup C_\ell | V - (C_1 \cup C_\ell))$ is a min-cut represented by a diagonal either A_1A_ℓ or B_1B_ℓ.*

Proof. Apply Lemma 3.8 with $C = C_1$ and $D + C_\ell$. □

Now we may put Theorem 3.7 and Corollary 3.9 together to show the existence of the min-cut representation. Let the W-representation consist of polygon $\mathcal{P}$. Assume wlog that A_1A_ℓ is a diagonal of $\mathcal{P}$ as described in Corollary 3.9; then there are no vertices of W mapped to the region bounded by the diagonals B_1A_1, A_1A_ℓ, $A_\ell B_\ell$, and the boundary section of the representing polygon between B_ℓ and B_1. Hence by the assumption that all outside cells are non-empty, we must have $B_1 = B_\ell$, whence it follows that all the B_i collide to B_1.

Now from $\mathcal{P}$, we build a new polygon representing all min-cuts as follows. From $\mathcal{P}$, we delete all diagonals A_iB_i with $2 \le i \le \ell - 1$; cut the C_1-side of diagonal A_1B_1, the C_ℓ-side of $A_\ell B_\ell$, and the $(W - C_1 - C_\ell)$-side of A_1A_ℓ. The remaining triangle $A_1A_\ell B_1$ contains neither vertices of W nor segments of diagonals. We replace this triangle by a rectangle $A_1A_\ell BB'$ and glue the three other pieces of $\mathcal{P}$ to the corresponding lines A_1B, $A_\ell B'$ and A_1A_ℓ. Finally, we add all diagonals A_iB and A_iB' and add a new atom $\{s\}$ to the outside cell adjacent to BB'. □□□

4 The proof

We prove in two steps update single polygons first and the tree-hierarchy next. In the first step, we only have to consider polygons that contain twin diagonal sides by Lemma 3.5. This part of the proof is based on the ideas of the cactus proof in Section 3.3, and in particular on Theorem 3.7.

In the second step, the key idea is that if we start by the possible *coarsest* subdivision of cuts into polygons (see Corollary 2.6), then there is a single polygon with twin diagonal sides. In fact, this requirement is necessary to obtain a valid hierarchy: a twin pair of diagonal sides could otherwise belong to two distinct polygons $\mathcal{P}_1$ and $\mathcal{P}_2$. Then the augmented polygons would contain a crossing pair of cuts and could not possess a tree-hierarchy. In contrast, under the assumption that there is only a single polygon containing twins, only minor changes may occur in the cactus defining the tree-hierarchy (Definition 1.4).

4.1 Polygons containing twins

In this section we fix a polygon $\mathcal{P}$ that contains diagonal sides $C_j \subset W$ with twin for $j = 1, \ldots, \ell$; the corresponding diagonals have end-vertices A_j and B_j as in Theorem 3.7. Let $\mathcal{S}(\mathcal{P})$ and $\mathcal{S}'(\mathcal{P})$ be as in Definitions 3.3 and 3.4.

First we prove that $\mathcal{R} = \bigcap \mathcal{S}'(\mathcal{P})$ contains no inside cells. Recall that $\mathcal{R}$ is a non-empty region by Lemma 3.5. By assuming that $\mathcal{R}$ is a non-inside region, the complement of $\mathcal{R}$ can be divided into regions $\mathcal{R}_i$ $(i = 1, 2, \ldots)$ (see Fig. 8–1); $i = 1$ if $\mathcal{R}$ is not separating. We prove that $\mathcal{R}$ is non-inside and hence the definition of the $\mathcal{R}_i$ makes sense; then another main fact we prove is that for each j, the diagonal A_jB_j may cross at most one of the regions $\mathcal{R}_i$.

Lemma 4.1. (i) $\bigcap \mathcal{S}''(\mathcal{P})$ *(and hence its subset* $\bigcap \mathcal{S}'(\mathcal{P})$*) contains no inside cells.* (ii) *Each* A_jB_j *crosses* $\mathcal{R}_i$ *for at most one* i.

Claim 4.2. *Let* C, D_1 *and* D_2 *be diagonal sides in* $\mathcal{S}(\mathcal{P})$. *Assume that* C *has a twin. Assume furthermore that both* D_1 *and* D_2 *cross* C. *Then* $C - D_1$ *and* $C - D_2$ *cannot be disjoint.*

Proof. We distinguish two cases. If $W - C$ is a twin of C (and consequently has to be extended by s), then $D_1 + s$, $D_2 + s$, $C + s$ and $V - C$ is four 6/5-near-mincut sides forming a 4-cycle (s in the common intersection), contradicting Lemma 2.13. Otherwise the twin of C is some $C' \neq \overline{C}$. But then $(D_1|W - D_1 + s)$, $(D_2|W - D_2 + s)$, $(C + s|W - C)$, and $(C'|\overline{C}')$ are all 6/5-near-mincuts forming a comb, contradicting Lemma 2.12. □

Proof of Lemma 4.1. (ii) is immediate by Claim 4.2, provided (i) holds so that the claim makes sense. As to (i), each cell $Z \in \bigcap \mathcal{S}''(\mathcal{P})$ must be adjacent to some diagonal A_jB_j and contained by its side C with twin. We prove that Z must be adjacent to either A_j or B_j, proving the lemma.

Assume Z is not adjacent to either of A_j and B_j. Then there are two diagonal sides D_1 and D_2, such that neither of them contains Z, both of them cross A_jB_j, and D_1 contains A_j, D_2 contains B_j. By Theorem 3.7, neither of them may have a twin. Hence by the definition of $\mathcal{R}$, both of them are complements of sets in $\mathcal{S}''(\mathcal{P})$ (i.e. their complement has to be extended by s). Hence by Claim 4.2, $C \cap D_1$ and $C \cap D_2$ cannot be disjoint. But then D_1 and D_2 are crossing: Z, A_j, B_j and $D_1 \cap D_2$ are in all distinct components of them. This concludes in the contradiction that A_jB_j, D_1 and D_2 form a 3-cycle. □

Corollary 4.3. *We may number regions* $\mathcal{R}_i$, $i = 1, \ldots, k$, *such that* $\mathcal{R}_i$ *covers the boundary of* $\mathcal{P}$ *between two cells* Z_{i-1} *and* Z_i; *furthermore* Z_0 *and* Z_k *are non-inside, the rest of the* Z_i *are separating (see Fig 8–1).*

Proof. By Theorem 3.7, the set of $j \leq \ell$ such that A_jB_j crosses $\mathcal{R}_i$ is a subinterval of $1 \ldots \ell$. Hence by Lemma 4.1, we may sort and renumber $\mathcal{R}_i$ according to the values of j such that A_jB_j crosses $\mathcal{R}_i$.

Let us prove the existence of Z_i, for $1 \leq i < \ell$. Let us consider the region ζ defined as the intersection of $\mathcal{R}$ (the complement of all $\mathcal{R}_i$) with the region between the last A_jB_j intersecting $\mathcal{R}_{i-1}$ and the first A_JB_J intersecting $\mathcal{R}_i$ $(J > j)$. By the definition of $\mathcal{R}$, only diagonals with twin may pass through this region. Now by Lemma 3.7 and the definition of j and J, these diagonals

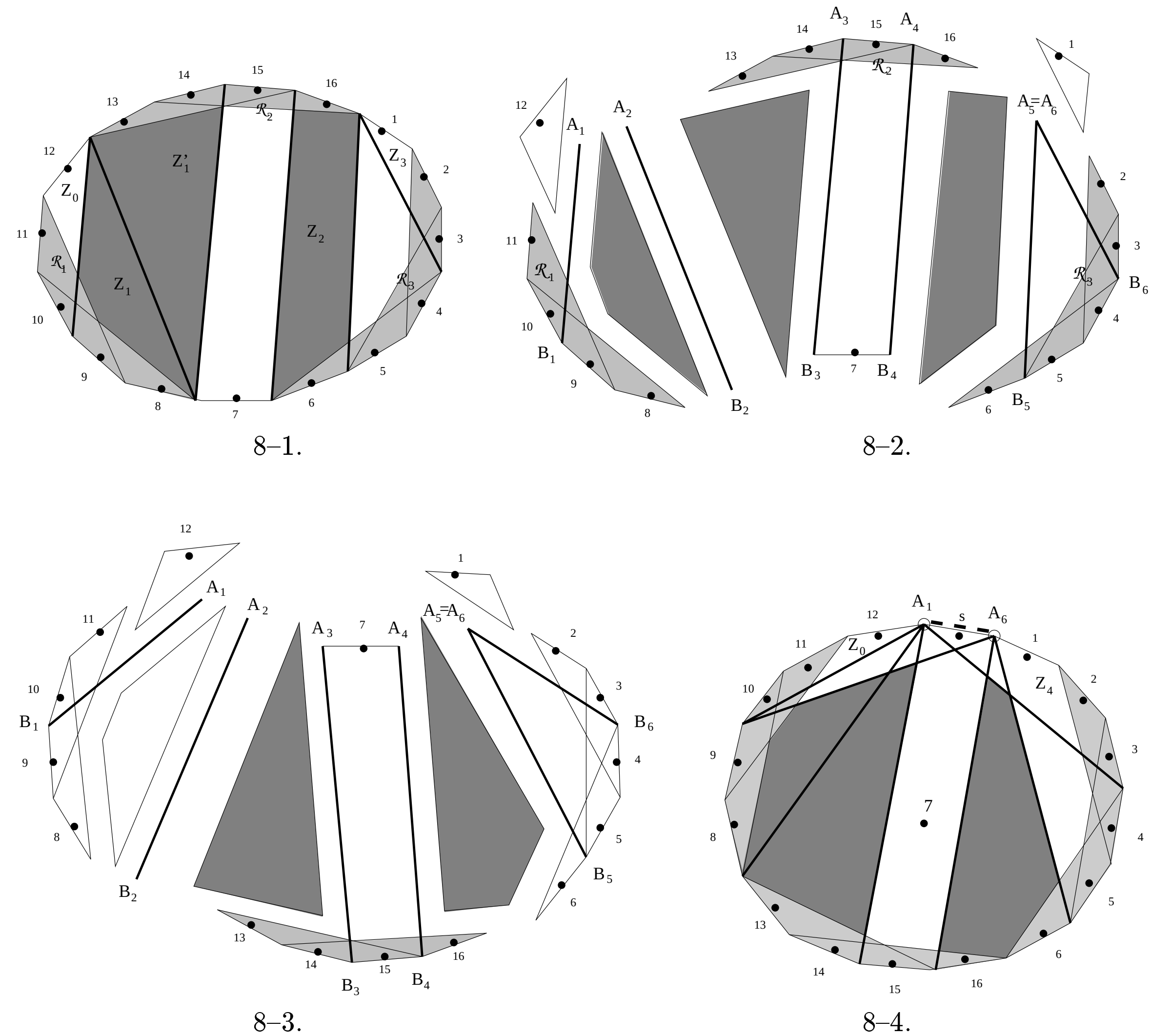

Figure 8: **1:** A polygon $\mathcal{P}$ containing twin pairs of diagonal sides; the corresponding diagonals are drawn thick. Light shaded areas are $\mathcal{R}_i$, $i = 1, 2, 3$. Dark shaded areas are separating cells between the $\mathcal{R}_i$; note that there are two choices Z_1 and Z_1' between $\mathcal{R}_1$ and $\mathcal{R}_2$. **2:** The $\mathcal{R}_i$ and the regions surrounded by diagonal sides crossing them are split by removing the separating cells. **3:** $\mathcal{R}_2$, together with the region between A_3B_3 and A_4B_4, is flipped around. **4:** The updated polygon. The new atom $\{s\}$ is adjacent to the edge A_1A_6 (dashed). The diagonal end-vertices A_j, $j = 1, \ldots, \ell = 6$ are redistributed to A_1, A_6, or both. Other diagonals are unchanged. Outside atom 7 of $\mathcal{P}$ becomes inside, the three separating (dark) cells become ambiguous.

may not cross any other diagonal. Hence ζ may only consist of separating cells; in particular, let $Z_i \in \zeta$ be one of them.

Let us prove the existence of Z_0 and (symmetrically) Z_k next. Let C_1 be the side of A_1B_1 not containing any other diagonals A_jB_j. If $\ell = 1$, i.e. there are no other A_jB_j, then Z_0 and $Z_1 = Z_k$ are found by using the argument with each side as C_1. By the next claim, we prove that C must intersect $\mathcal{R}$, whence $C - \mathcal{R}_1$ is non-empty. But then $C - \mathcal{R}_1$ must be a single cell Z_0, since no diagonal may pass through it. Z_0 may not be inside by Lemma 4.1.

Claim 4.4. *For all $j \leq \ell$, $\mathcal{R} = \bigcap \mathcal{S}'(\mathcal{P})$ has cells on both sides of A_jB_j.*

Proof. Follows by applying Lemma 3.5 with $\mathcal{S}' = \mathcal{S}'(\mathcal{P}) + C_j$ and $\mathcal{S}' = \mathcal{S}'(\mathcal{P}) + C_k$, where C_k is a twin of C_j. □

Now we describe the new representing polygon in terms of the sequence of its outside atoms. Let $X_1^{(i)}, \ldots, X_{r(i)}^{(i)}$ be the sequence of outside atoms in $\mathcal{R}_i$, from Z_{i-1} to Z_i.

Theorem 4.5. *Let $\mathcal{C}$ be the collection of cuts of G whose intersection with W is represented in $\mathcal{P}$. The following sequence of outside atoms defines the representation of all cuts of $\mathcal{C}$:*

$$\{s\}, Z_0, X_1^{(1)}, \ldots, X_{r(1)}^{(1)}, X_1^{(2)}, \ldots, X_{r(k)}^{(k)}, Z_k \; .$$

This theorem can also be expressed in terms of cutting and gluing $\mathcal{P}$, as we did it with the cactus representation. We cut $\mathcal{P}$ at all separating cells of $\mathcal{R}$ to separate the $\mathcal{R}_i$, see Fig. 8–2. Then in order to form the consecutive sequence of outside atoms as in the theorem, we possibly flip some $\mathcal{R}_i$ and the region ζ enclosed by the A_jB_j crossing $\mathcal{R}_i$ (Fig. 8–3). After flipping, we exchange A_j and B_j so that $B_j \in \mathcal{R}_i$. Finally (Fig 8–4), we remove all formerly outside atoms within the regions $\zeta - \mathcal{R}_i$ (these atoms can easily be shown to be inside in the new representation), glue all pieces together, and close the boundary with a new edge A_1A_ℓ. The A_j are redistributed to A_1 or A_ℓ (or both), depending on which side of A_jB_j has to be extended by s. The new atom $\{s\}$ will be the outside atom adjacent to A_1A_ℓ.

Proof. We prove (i-iii) of Theorem 2.4 for the above sequence of atoms. We know that (iii) holds for the entire collection of cuts to be represented (by Lemma 2.13). To prove (i), let us consider a diagonal side that has to be extended by s and has no twin. This diagonal side covers the entire polygon except for some $\mathcal{R}_i$. Consequently any cut side $s \in C$ whose corresponding diagonal side has no twin in $\mathcal{P}$ contains all the atoms except for some $X_j^{(i)}$ for a fixed i—including $\{s\}$.

Next we show (i) of Theorem 2.4 cuts corresponding to diagonals A_jB_j ($j = 1, \ldots, \ell$). Since both sides of these diagonals may potentially have to be extended by s, we consider both possible corresponding cuts in G. Requirement (i) holds for these cuts as well, since in $\mathcal{P}$, these diagonals cut $Z_0, X_1^{(1)}, \ldots, Z_k$ into two consecutive parts; we may complete these cuts both by including $\{s\}$ and by not.

Hence all is left to do is to prove (ii). Let us consider a pair of cut sides C_1 and C_2, let C_1' and C_2' be the corresponding diagonal sides of $\mathcal{P}$. We prove that C_1 and C_2 satisfy (ii). First assume that $C_1' \cap C_2'$ does not contain outside atoms of $\mathcal{P}$. Then $C_1' \cap C_2' = \emptyset$, since (ii) holds for $\mathcal{P}$. Thus $C_1 \cap C_2$ is either empty or $\{s\}$. As $\{s\}$ is a new outside atom, we are done in both cases. Now we assume $C_1' \cap C_2'$ contains outside atoms of $\mathcal{P}$ and prove by distinguishing three cases.

1. If one of C_k' ($k = 1, 2$) is contained by a region $\mathcal{R}_i$, then C_j' contains exactly the same inside atoms of the new system as of the old, namely $X_1^{(i)}, \ldots, X_{r(i)}^{(i)}$. We are done.

2. All diagonal sides other than those in case 1 contain Z_0 or Z_ℓ. If one of C'_k contains both, then $C_1 \cap C_2$ also contains one of them. We are done again.

3. The only remaining case is when $Z_0 \in C'_1 - C'_2$ and $Z_\ell \in C'_2 - C'_1$. Then both C'_1 and C'_2 are sides of the diagonals $A_j B_j$ $(j = 1, \ldots, \ell)$. The corresponding cut sides C_1 and C_2 are either disjoint, or contain $\{s\}$. The proof is completed. $\square$

4.2 The tree-hierarchy

Let $\mathcal{K}$ be the cactus defining the tree-hierarchy of the W-representation (see Definition 1.4). We prove here that $\mathcal{K}$ defines the tree-hierarchy of the new representation as well. We use the assumption that the initial W-representation contained no pair of outside connecting atoms. Under this assumption, we prove that there is a single polygon containing twins. Then we assume that this polygon is updated as in the previous subsection, and the rest of the polygons are updated in the trivial way.

In fact, the cactus $\mathcal{K}$ has to be slightly modified. First of all, there may be new atoms $\{s\}$ so that the cycles have to get increased. Second, there may be a single new polygon consisting of the cut $(s|V-s)$. A length two cycle $\{s, V_s\}$ has to be added to $\mathcal{K}$, with $V-s$ contracted to all nodes with minimal atoms containing s.

Lemma 4.6. *Let Z_1 and Z_2 be the connecting atoms of two polygons $\mathcal{P}_1$ and $\mathcal{P}_2$ (with $Z_1 \cup Z_2 = W$). Let $C \in \mathcal{S}(\mathcal{P}_1)$ be a diagonal side not containing atom Z_1. Let $D \in \mathcal{S}(\mathcal{P}_2)$. Then either C and D are twins, or D contains Z_2.*

Proof. By $Z_1 \subseteq W - C$ and $Z_1 \cup Z_2 = W$, we get that $C \subseteq Z_2$. Since Z_2 is an atom if D, if D does not contain Z_2, then D is disjoint of Z_2 and hence of C. Thus C and D are twins as required. $\square$

Let a polygon $\mathcal{P}$ of the W-representation be called *bad* if it contains a diagonal side with twin (in, possibly, another polygon). Otherwise, let $\mathcal{P}$ be *good*. Now we complete our goal in the special case when there is at most one bad polygon. Let Z_1, Z_2 be as above. Let $\mathcal{P}_1$ be good. Assume that by updating $\mathcal{P}_1$, s is *not* added to Z_1. Since for $\mathcal{P}_1$, only the second possibility of Lemma 4.6 can hold, for all $D \in \mathcal{S}(\mathcal{P}_2)$, $Z_2 \subseteq D$. But $\bigcap \mathcal{S}(\mathcal{P}_2) \supseteq Z_2$ implies that $\mathcal{P}_2$ is good and s is added to Z_2 in $\mathcal{P}_2$.

We prove in two steps that there cannot be more than one bad polygons. The following lemmas complete the proof of Theorem 2.9.

Lemma 4.7. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two bad polygons with either both containing a twin pair or with a twin pair between $\mathcal{P}_1$ and $\mathcal{P}_2$. Then all polygons $\mathcal{P}_3$ are bad on the shortest path of the cactus $\mathcal{K}$ defining the tree-hierarchy; furthermore either $\mathcal{P}_1, \mathcal{P}_3$ or $\mathcal{P}_2, \mathcal{P}_3$ contains a twin pair.*

Proof. For any $\mathcal{P}_3$ on the above path, there must be a diagonal $(C|W-C)$ of $\mathcal{P}_3$ separating atom Z where the path enters from atom Z' where the path leaves $\mathcal{P}_3$. Let $C_1 \in \mathcal{S}(\mathcal{P}_1)$ and $C_2 \in \mathcal{S}(\mathcal{P}_2)$ form a twin pair. Since $C_1 \cap C_2 = \emptyset$, we must have

$$C_1 \subset Z \subset C \quad \text{and} \quad W - C \supset Z' \supset C_2 \ .$$

Furthermore, either C or $W-C$ has to be extended by s; if this side is C, then C and C_2 are twins, otherwise $W-C$ and C_1. $\square$

Lemma 4.8. *In the possible coarsest subdivision of cuts into polygons, there is a single polygon of the W-representation containing all diagonal sides with twin.*

Proof. We derive a contradiction from the assumption that there are two bad polygons $\mathcal{P}_1$ and $\mathcal{P}_2$ such that either both of them contains a pair of twins, or there is a twin pair $C_1 \in \mathcal{S}(\mathcal{P}_1)$ and

$C_2 \in \mathcal{S}(\mathcal{P}_2)$. By Lemma 4.7, we may assume that $\mathcal{P}_1$ and $\mathcal{P}_2$ are incident in the tre–hierarchy with connecting atoms Z_1 and Z_2. Since either Z_1 or Z_2 must be inside, by Lemma 4.1, either $Z_1 \notin \bigcap \mathcal{S}''(\mathcal{P}_1)$, or the same holds for $\mathcal{P}_2$ and Z_2. By assuming the first possibility, we derive a contradiction that there is a twin pair neither in $\mathcal{P}_2$ nor between $\mathcal{P}_1$ and $\mathcal{P}_2$.

Let us assume that there is a $C \in \mathcal{S}''(\mathcal{P}_1)$ with $Z_1 \subseteq W - C$. Let $D \in \mathcal{S}(\mathcal{P}_2)$ be arbitrary; C and D are not twins by the definition of C. Hence by Lemma 4.6, $Z_2 \subseteq D$, for all $D \in \mathcal{S}(\mathcal{P}_2)$. On one hand, this implies that any two elements of $\mathcal{S}(\mathcal{P}_2)$ intersect, i.e. there is no twin pair in $\mathcal{S}(\mathcal{P}_2)$. On the other hand, for $D \in \mathcal{S}(\mathcal{P}_2)$, $D \supseteq Z_2 \supseteq W - Z_1 \supseteq Z$, for any atom $Z \neq Z_1$ of $\mathcal{P}_1$. Hence for $D' \in \mathcal{S}(\mathcal{P}_1)$, $Z \subset D' \cap D$ for some atom other than Z_1 contained by D'. This proves there is no twin pair between $\mathcal{P}_1$ and $\mathcal{P}_2$. □□□

5 The algorithm

First we show that the proof steps of Section 4 can be performed algorithmically. The steps below are all deterministic; randomization is only needed for efficiently finding the cuts. Then in Section 5.2, we describe an $O(n^2 \log n)$-time algorithm that transforms the representation of cuts found by the Karger–Stein algorithm to our representation. Notice that extra ideas are needed in that subsection, since the explicit list of all cuts may not be produced (see Section 1.3).

Our goal is to turn the proof steps of Section 4 to an amortized $O(n \log n)$-time algorithm. The main idea of this section is to build the sequence of outside atoms satisfying Theorem 2.4 first. Then if the lists of all outside atoms are given, we build polygons and find the positions of the inside atoms in this polygon. It will turn out that the position of a single inside cell can be found in $O(n \log n)$ time. Hence if we wanted to come up with the polygons of all W-representations for a sequence $W_1 = \{w\} \subset W_2 \subset \ldots \subset W_n = V$, we should spend $\binom{n}{2} O(n \log n)$ time. Instead, if we locate inside cells only at the end, we may amortize this cost to each individual increment step.

Notice that we have to use geometric algorithms sparingly: although by Theorem 2.2, the total edge size of the polygons to $O(n)$, within a single polygon of size $\Omega(n)$, the total number of cells may be $\Omega(n^4)$. Hence geometric search structures as for example sweeping through all cells cannot efficiently handle the representation. We build the following structure to algorithmically handle polygons:

Definition 5.1. For the polygon representation, we maintain the following information:

1. an auxiliary data structure (probably only a list) of the possible $\binom{n}{2}$ represented cuts;
2. for each polygon, the coordinates of the k polygon-vertices;
3. for each polygon-vertex, the list of cuts with representing diagonal starting at this vertex. This list also includes the other diagonal-endvertex and is sorted according to that;
4. the coordinates of all non-empty atoms; and
5. for each atom Z, a list of $O(n)$ diagonals of the polygon of Z, including all those with minimal side *containing* Z (which may or may not actually *bound* the cell of Z);
6. the cactus defining the tree-hierarchy.

Since we may assume that the coordinates of each polygon-vertex are integers between 1 and n, and since the position of each atom can be a point close to the intersection of two diagonals, the total size of this data structure is $O(n^2)$. □

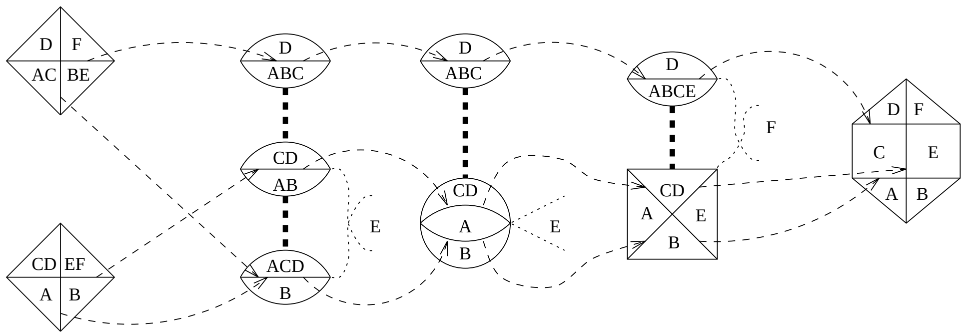

Figure 9: Increasing the representation by two new vertices E and F. In this particular case, the list of represented cuts (data 1 of Definition 5.1) are given by the two smaller polygon representations (see Section 5.2). Thick dashed lines denote connecting atoms; dashed arrows are cut pointers to/from the initial list.
From left to right: The two representations; The $ABCD$-representation, where CD and B are twins (dotted lines indicate the position of E for these two diagonals); The single bad polygon is formed; The $ABCDE$-representation; and The final representation.

Next we give the main steps of our $O(n \log n)$-time algorithm to augment the W-representation by a vertex s (see Fig 9). The input contains the tree-hierarchy as well as for each polygon of the W-representation, the sequence of outside atoms and the list of diagonals starting between pairs of atoms. Furthermore for each vertex v, the input contains a list of $O(n)$ cuts including all the minimal ones containing v. This list can be divided to polygons to form data 5 above. The output is the $(W + s)$-representation, described with a data structure of the same form.

Given this algorithm, we build the actual polygons and determine data 4–5 in Definition 5.1 at the very end, when $W = V$. We will show that this last step requires $O(n \log n)$ time per atom. Hence the final time bound is $O(n^2 \log n)$.

In the updating procedure of Section 4, the knowledge of the possible *coarsest* subdivision of polygons is required. However, we assume that the system of polygons is given as the possible *finest* subdivision (for the possible subdivisions, see Corollary 2.6). The required coarsest subdivision is then given as a partition the system of polygons into the largest possible subsets that can be joined to a single polygon. From now on, we refer to such a collection as a polygon, although we do not physically join them. The importance of this assumption is that flipping regions around separating cells can be performed in constant time.

Algorithm 5.2 (Increasing the representation).

1. Identify the single bad polygon $\mathcal{P}$ (containing twin diagonal sides), if there is one.

2. For all other (good) polygons, find the atom of s as the intersection of minimal cuts containing s. Update the atoms in the tree-hierarchy; also construct the possible new polygon representing $(s|V - s)$.

3. Find all diagonals A_jB_j, $j = 1, \ldots, \ell$ of $\mathcal{P}$ as in Theorem 3.7.

4. For each sub-polygon $\mathcal{P}_0$ of $\mathcal{P}$, find the A_jB_j contained by $\mathcal{P}_0$. Find the boundary segment of $\mathcal{P}_0$ belonging to $\mathcal{R}$. Cut $\mathcal{P}_0$ into $\mathcal{P}_0 \cap \mathcal{R}$ (containing no diagonals other than the A_jB_j) and $\mathcal{P}_0 - \mathcal{R}$ (with all other diagonals).

5. Form the sequence of outside atoms as in Theorem 4.5 and build the sequence of diagonals as in 3 of Definition 5.1. This can be done by gluing all sub-polygon pieces. These pieces contain all diagonals except the A_jB_j. The new diagonals corresponding to A_jB_j can then be inserted, in $O(\log n)$ time each.

End.

The main routine we need in Algorithm 5.2 is to algorithmically perform Lemma 3.5. We describe the following procedure in the next subsection.

Algorithm 5.3 (Cyclic scan). The *input* is a polygon $\mathcal{P}$ (or a logically joined collection of polygons) of the W-representation and a 3-cycle free set of diagonal sides $\mathcal{S}$. The polygons have a total n' edges and are given by 1 and 3 of Definition 5.1. $\mathcal{S}$ is given by pointers to the diagonal lists of the polygons; $|\mathcal{S}| = O(n')$. If $\mathcal{S}$ contains disjoint pairs, the *output* is a pair $C_1, C_2 \in \mathcal{S}$ with $C_1 \cap C_2 = \emptyset$ and such that $C_1 \cup C_2$ is minimal. Otherwise, the *output* is the list of minimal elements of $\mathcal{S}$; if the polygon-vertices are given by coordinates, then a point in $\bigcap \mathcal{S}$ is also found. The running time is $O(n' \log n)$.

By the above procedure we may complete the description of the incomplete steps 1, 3 and 4. Let first $\mathcal{S}$ include all minimal cut sides containing s; note that this list is part of the input. With this $\mathcal{S}$, we may perform the "goodness-test" of step 1. For the bad polygon, we can output the first and last diagonals A_1B_1 and $A_\ell B_\ell$ with twins; then since the A_jB_j are precisely those parallel to A_1B_1 and $A_\ell B_\ell$, they can be found by a binary search at each vertex A between A_1 and A_ℓ. This completes step 3. Finally we remove all A_jB_j from $\mathcal{S}$ and find all minimal elements; thus we may identify $\mathcal{R}$ as required in step 4.

We may also build the final polygons by Algorithm 5.3. We consider each polygon given by the list of outside atoms. We give suitable coordinates for the polygon-vertices. For the atom of $v \in V$, we set $\mathcal{S}$ to be the list including all minimal cut sides containing v, as given in the input. Then Algorithm 5.3 finds data 4–5 of Definition 5.1.

5.1 Cyclic scan (Algorithm 5.3)

We describe our algorithm such that it aborts when a disjoint pair C_1, C_2 is found in $\mathcal{S}$. Then a pair with $C_1 \cup C_2$ minimum can be found by finding a minimal set contained by C_1 and one within C_2. The search for these two sets can be performed in $O(n \log n)$ time. Also notice that a point in $\bigcap \mathcal{S}$ can be found by linear programming, using the diagonals to define inequalities. The existence of a solution is guaranteed by Theorem 2.4, since $\mathcal{S}$ is 3-cycle free. Linear programming in 2 dimensions can be performed in linear time [6]. The special structure of our problem could also yield a less complicated $O(n \log n)$-time procedure.

We also assume that for each polygon-vertex, there is a single diagonal of $\mathcal{S}$ starting at that vertex. If this assumption fails, the two elements of $\mathcal{S}$ either contain one another, or are disjoint.

Now the algorithm starts with some diagonal A_0B_0 corresponding to a diagonal side $C_0 \in \mathcal{S}$. Then starting from A_0, the polygon-endvertices A in the complement of C_0 are scanned in cyclic order. For diagonals AB, continue as long as the sequence of B's is in cyclic order in C_0. If for an AB, B is not in cyclic order, that diagonal side is not minimal and can be removed. And if $B \notin C_0$, then AB and A_0B_0 are disjoint; abort.

Now assume we scanned all vertices in the complement of A_0B_0. Then we move A_0 to the first A_1 scanned with A_1B_1 not removed from $\mathcal{S}$. We continue scanning the complement of the corresponding C_1 from B_0 on. We do this procedure with a sequence of diagonals A_1B_1, $A_2B_2, \ldots$, as long as we reach A_0B_0 again.

Lemma 5.4. *If the above procedure does not abort, all remaining elements of $\mathcal{S}$ are minimal and $\mathcal{S}$ contains no disjoint pair.*

Proof. Assume that sides C_i and C of diagonals A_iB_i and AB in $\mathcal{S}$ are counterexamples to the claim: they are either disjoint or $C_i \subset C$. In both cases, B_i is in C_i. We may assume that the counterexample is such that B is as close to B_i in C_i as possible. When the complement of C_i is scanned, we may not reach AB, since otherwise it is either removed or the procedure aborts. However, there must be an A_jB_j such that A is scanned in the complement of C_j. This is possible only if B_j is between B and B_i in C_i and A is in the complement of C_j. Then by assumption, AB and A_jB_j are non-disjoint and $C_j \not\subset C$, and the same holds for the pair C_i, C_j.

Now we distinguish four cases, depending whether A_jB_j crosses AB and A_iB_i or not. Notice that A_iB_i does not cross AB, whence A_jB_j cannot cross both of them. This eliminates one case. If A_jB_j does not cross A_iB_i, then $C_i \subset C_j$, which is impossible. Finally, if A_jB_j does not cross AB, then both A and B are in the complement of C_j, when either C and C_j are disjoint or $C \subset C_j$ should hold. We reached a contradiction. □

5.2 Applying the Karger–Stein algorithm

The Karger–Stein algorithm builds a binary tree of graphs, by recursively contracting graphs as we move farther from the root. To achieve the very efficient running time, it is crucial that for an intermediate graph G' with n' vertices, we spend only $O(n'^2)$ time for contracting G'. But in that case if $n' = o(n)$, we are not allowed to carry direct pointers to vertices of the *original* graph. The idea of our algorithm is to build the representation up the binary tree given by the Karger-Stein algorithm: in each step, the cut representations of the two different contracted graphs are merged as the vertices of the graph get "unwrapped".

First we describe the implicit representation of cuts found by the Karger–Stein algorithm. It consists of a binary tree $\mathcal{T}$ of graphs. The algorithm builds $\mathcal{T}$ as follows. For a graph G_0 with n_0 vertices on the current leaf level of $\mathcal{T}$, the two new children will contain graphs with $n_0/\sqrt{2}$ vertices. These two graphs arise by applying the (randomized) contraction algorithm of Karger [13] twice for G_0. The final leaves of $\mathcal{T}$ contain graphs with two vertices, or equivalently, a two sides of a cut, each contracted to a single vertex. To achieve the very efficient running time, it is crucial that for an intermediate graph G_0, we may spend only $O(n_0^2)$ time for contraction. But in that case if $n_0 = o(n)$, we are not allowed to carry information on the *original* graph. Hence at a vertex v of G_0, all we know is the set of vertices in the parent of G_0 that are contracted to v. Vertices of a cut can be "unwrapped" by traversing the path up to the root.

Our algorithm recursively merges the representations as we reconstruct graphs, moving up the binary tree $\mathcal{T}$ produced by the Karger–Stein algorithm (similarly to the cactus-algorithm of [15]). Let G_0 be a graph in $\mathcal{T}$; let G' and G'' be its two children. Assume we have a representation of the collection $\mathcal{C}'$ of all near-mincuts found at leaves below G', and another for $\mathcal{C}''$ below G''. We build the polygon representation of $\mathcal{C}' \cup \mathcal{C}''$.

We start by the representations of $\mathcal{C}'$ and $\mathcal{C}''$ as a data structure in 1 of Definition 5.1 (see Fig. 9). For each vertex s, an $O(n_0)$-size lists of diagonal sides including the minimal ones that contain s can be given by merging the two corresponding lists in $\mathcal{C}'$ and $\mathcal{C}''$. Then we run the algorithm of the previous section in $O(n_0^2 \log n)$ time. Our running time for each G_0 is bounded by the running time of the Karger–Stein algorithm for the two randomized contraction steps.

6 Splittings that preserve near-mincuts

In this section we give an efficient splitting-off algorithm. We also define the notion of splitting-off by preserving near-mincuts; we prove that splittable pairs can be found by the polygon representation. Our result extends Lovász' Splitting Theorem (1.2).

Let us consider the algorithmic side of Lovász' Theorem first. In all previous work [8, 10], a vertex s is split off algorithmically in the following way: edge pairs us, vs are chosen and the maximum value w is computed such that splitting with weight w preserves the value of c'. Now w is found by basically a flow computation. Let $c_{u,v}$ be the minimum value of a cut separating u and v from s other than $(s|V-s)$; then it is easy to see that $w = (c_{u,v} - c')/2$.

It is obvious from the above discussion that near-minimum cuts play a key role in the edge splitting problem. In graphs with integer edge weights, us and vs is splittable with weight at least one (i.e. $c_{u,v} > c' + 1$) precisely if u and v are not simultaneously separated from s by a minimum or minimum+1 cut.

We obtain an improved splitting-off algorithm by considering not only minimum+1, but constant times minimum cuts as well. Instead of finding splittable weights for single edge pairs one by one, we may then validate that a *sequence* of edge pairs is splittable, without decreasing the mincut value. The validation goes as follows: let a sequence of edge pairs e_i, f_i be given, with weights w_i, for $i = 1, \dots \ell$. By splitting this entire sequence of edge pairs, the value of a cut may decrease by $2\sum w_i$, in the worst case. Thus by knowing all cuts with value less than $c + 2\sum w_i$, we can validate in one step whether the sequence is splittable.

Definition 6.1 (δ-Splitting). Let G be a graph with a distinguished vertex s. Let $V-s$ have connectivity c'. Let us consider pairs of cuts $(C|\overline{C})$ and $(C-s|\overline{C}+s)$ with $s \in C$, $C \neq s$, such that at least one of the two cuts have value less than $\delta c'$. We say that a pair of edges us, vs is *δ-splittable* with weight w, if the minimum cut value of none of the above pairs of cuts decrease.

Now we derive our splitting-off algorithm. Our next theorem states the existence of δ-splittable pairs, for $\delta \le 6/5$. More importantly, by computing the representation of δ-near-mincuts only once, we show that all edge weights incident to s can be paired up to δ-splittable pairs. From these edge pairs, a total weight of $c\delta'$ pairs (or $\lfloor c\delta' \rfloor$ pairs with integer weights) are $(\delta - 2\delta')$-splittable by definition. Hence in one step, we split $O(c)$ weight from s and split s off in $O(d(s)/c)$ steps, for suitable constants $\delta \le 6/5$ and $\delta' > 1$.

The running time analysis now is as follows. By setting $\delta = 1 + 1/\log n$, the representation of δ-near-minimum $(V-s)$-cuts can be found in $\tilde{O}(n^2)$ time. Then the entire running time will be $\tilde{O}(n^2 d(s)/c')$. A slight improvement over this result is given in Section 6.1.

Theorem 6.2 (δ-Splitting). *For $\delta \le 6/5$, there are $O(n)$ edge pairs $u_i s$, $v_i s$, such that each pair is δ-splittable with weight w_i, and that these pairs split s off. Furthermore, if all edge weights are integral and $d(s)$ is even, then pairs $u_i s, v_i s$ exist with all w_i integer. The edge pairs can be found in $O(n^2 \log n)$ time, given the representation of all δ-near-minimum $(V-s)$-cuts.*

Corollary 6.3. *For an undirected graph with distinguished vertex s, s can be split off with fractional weights as in Lovász' Theorem 1.2, in time $\tilde{O}(n^2 d(s)/c')$. Here c' is the connectivity of $V-s$. If all edge weights are integral, $d(s)$ is even and $c' = \Omega(\log n)$, then s can be split off with integer weights within the same time bound. And if $c' \ge 6$, then the running time is $\tilde{O}(n^{2.4} d(s))$.*

Proof. The only statement left to be justified is that splitting-off with integer weights is possible. Notice that in each step, a total weight $\lfloor c\delta' \rfloor$ of edges can be split off. If $c' = \Omega(\log n)$, with the choice $\delta = 1 + 1/\Theta(\log n)$, we may split s off in $O(d(s)/c')$ steps. Otherwise, we have to set $\delta = 6/5$ and then at least one splittable pair is found in each step. □

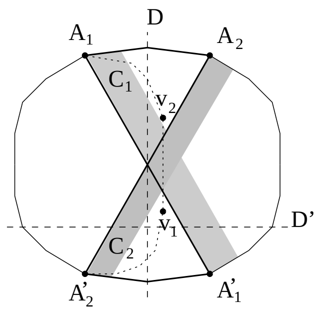

Figure 10: In the proof of the δ-Splitting Theorem, the two consecutive crossing sets C_1 and C_2 contain a maximal subset of Γ subject to having weight at most $\frac{1}{2}d(s)$. The dotted line surrounds the convex hull of vertices of $\Gamma \cap (C_1 - C_2 \cup C_2 - C_1)$, A_1 and A_2'. D is a possible cut that constrains the splittable weight. There is no cut D', since it forms a 3-cycle containing v_1.

Proof of Theorem 6.2. Let Γ denote the set of vertices incident to s. Let the *weight* w of a subset of Γ be the sum of the weights of edges entering from s. Let us build the representation of δ-near-minimum $(V - s)$-cuts. Now we repeat the following algorithm until Γ becomes empty.

Algorithm 6.4 (Finding a δ-splittable pair).

1. Let $\mathcal{P}$ be a polygon with not all elements of Γ belonging to the same atom.

2. For each polygon-vertex A_i, let C_i be the clockwise maximal diagonal side such that $w(C_i \cap \Gamma) \le \frac{1}{2}d(s)$; let its other end-vertex be A_i'. These diagonals can be found by binary search, using 3 of Definition 5.1.

3. Find all C_i with $C_i \cap \Gamma$ maximal. Run the cyclic scan routine (Algorithm 5.3) with $\mathcal{S} = \{\overline{C}_1, \overline{C}_2, \ldots\}$, and with the modified definition of containment (i.e. C_i is removed from $\mathcal{S}$ if $C_i \cap \Gamma \subset C_j \cap \Gamma$, for some j. Since $w(C_i \cap \Gamma) \le \frac{1}{2}d(s)$, there is no disjoint pair of $C_i \cap \Gamma$.

4. If there is a single maximal $\mathcal{C}_i$, pair all vertices in $C_i \cap \Gamma$ to vertices in the complement arbitrarily. If there are more than one maximal $\mathcal{C}_i$ but they are pairwise disjoint, then pair vertices in pairs of distinct $C_i \cap \Gamma$ arbitrarily.

5. If the previous two cases do not hold, there is a pair C_1 and C_2 of maximal elements with $C_1 \cap \Gamma$ and $C_2 \cap \Gamma$ crossing. Assume wlog that $A_2 \in C_1$ and $A_1' \in C_2$ (see Fig 10). Let us pick C_1 and C_2 such that A_2 is as close to A_1 as possible. Build the convex hull of A_1, A_2', and all vertices of Γ in the symmetric difference of C_1 and C_2.

6. Let $v_1 \in C_1 - C_2$ and $v_2 \in C_2 - C_1$ be the unique vertices of Γ such that v_1v_2 is the edge of the convex hull crossing the two diagonals A_1A_1' and A_2A_2' (see Fig. 10). Notice that both $C_1 - C_2$ and $C_2 - C_1$ contains vertices of Γ, since $C_1 \cap \Gamma$ and $C_2 \cap \Gamma$ are crossing.

7. Find the minimum w_0 of $w(C)$ for diagonal sides C containing both v_1 and v_2. For each possible A, find the minimizing diagonal by binary search. Split sv_1 and sv_2 by weight $\min\{w(v_1), w(v_2), w_0 - d(s)/2\}$. Update w and Γ.

End.

We prove the correctness of this procedure with the following simple lemma. This lemma itself implies that the edge pairs found in step 4 are δ-splittable. It is slightly more difficult to prove the same for pairs found in step 7. The last two proof steps are bounding the number of edge pairs found, and proving the running time.

Lemma 6.5. *A pair of edges us, vs is δ-splittable with weight $w_{u,v}$ iff there is no set $D \subset V - s$ containing both u and v with $w(D) < \frac{1}{2}d(s) + w_{u,v}$. The maximum $w_{u,v}$ is a non-zero integer if all edge weights are integral and $d(s)$ is even.*

Proof. Let us consider the two cuts $(D|\overline{D})$ and $(D + s|\overline{D} - s)$ corresponding to D. Notice that Definition 6.1 has to be validated only for sets D containing both u and v: the value of no other cuts may change. Hence us, vs is splittable with weight $w_{u,v}$ iff for all D containing both u and v, $\min\{d(D), d(\overline{D} - s)\}$ does not decrease. Now the claim follows by expressing $d(\overline{D} - s)$ as $d(D) - w(D) + w(\overline{D} - s) = d(D) + d(s) - 2w(D)$. □

Lemma 6.6. *Edges v_1s and v_2s are δ-splittable with weight as in step 7. This weight is non-zero and integer if all edge weights are integral and $d(s)$ is even.*

Proof. We use Lemma 6.5. Let D contain v_1 and v_2 with $w(D)$ minimum; we have to prove that $w(D) > \frac{1}{2}d(s)$. Hence we may assume D is one of the C_i as in step 3 of Algorithm 6.4. It is easy to see that D must contain at least two vertices of A_1, A'_1, A_2 and A'_2. We distinguish the following cases:

- If both A_1 and A'_1 (or A_2 and A'_2) are in D, then D contains either $C_1 \cap \Gamma$ (or $C_2 \cap \Gamma$), or $\Gamma - C_1$ (or $\Gamma - C_2$). In the first case, $w(D) \geq w(C_1) + w(v_1) > w(C_1)$; by the definition of C_1, $w(D) > \frac{1}{2}d(s)$ and the weight is non-zero. Similarly in the second case, $w(D) \geq w(\overline{C}_1) + w(v_2) > \frac{1}{2}d(s)$.
- If D contains A'_1 and A_2 does not contain A_1 and A'_2, and has $w(D) \leq \frac{1}{2}d(s)$, then we are in contradiction with the assumption in step 5 of Algorithm 6.4 that A_2 is as close to A_1 as possible.
- If D contains A_1 and A'_2, then by the choice of v_1 and v_2, D contains all vertices either in $(C_1 - C_2) \cap \Gamma$ or in $(C_2 - C_1) \cap \Gamma$. D also contains $\overline{C}_1 \cap \overline{C}_2$, hence D contains $\overline{C}_1 \cap \Gamma$ or $\overline{C}_2 \cap \Gamma$. We may then follow the first case.
- Finally, if D contains A_1 and A_2 (or A'_1 and A'_2), then D forms a 3-cycle with C_1 and $\overline{C}_2$ (Definition 2.3). This is impossible by Lemma 2.13. □

Lemma 6.7. *We have to run Algorithm 6.4 $O(|\Gamma|) = O(n)$ times before $\Gamma = \emptyset$. The number of edge pairs found is bounded by the same amount.*

Proof. In the above procedure, we either entirely remove a vertex from Γ, or create a new set D with $w(D \cap \Gamma) = w(\Gamma - D) = d(s)/2$. Whenever such a set D is created, it will preserve the above property for any further splittings. Since there are at most $|\Gamma|/2$ such sets D, in at most $3/2 \cdot |\Gamma|$ steps we either remove all pairs or stop with one single atom containing Γ, and change to another polygon. Altogether, we find $O(n + |\Gamma|) = O(n)$ edge pairs. □

Lemma 6.8. *Each call to Algorithm 6.4 takes $O(n \log n)$ time.*

Proof. First note that testing whether an atom is contained by a diagonal side can be done in constant time, given the coordinates of the atom and the diagonal. Next we analyze the only non-trivial steps 3 and 5. Step 3 is a call to Algorithm 5.3 that runs in time $O(n \log n)$. Finally, in step 5, a convex hull has to be computed. If we pre-process each polygon such that for each polygon-vertex, the lists of atoms is sorted by angle, we can easily compute the convex hull in $O(n)$ time. The pre-processing time $O(n^2 \log n)$ is bounded by the time to find the polygons. □

The proof of Theorem 10 is completed by Lemmas 6.7 and 6.8. □□□

Remark 6.9. Notice that even if all edge weights are integral, all we know of the splittable weights is that they are 1/2-integral. Clearly there is no integer splitting-off if $d(s)$ is not even; however the existence of a single splittable pair is still in question. Even in the setting of Lovász' Theorem, extra conditions are needed to ensure the existence of a single (weight 1) pair; for example, it is necessary that $d(s) \neq 3$ [18, 7]. For δ-splittable pairs, the requirements are even stronger. Consider a $(2\ell+1)$-wheel graph with weight ℓ on its outside cycle (Fig. 4). It has no pair adjacent to the middle vertex which could be split (by weight 1) by preserving all minimum and minimum+1 cuts.

6.1 Graph orientation and improved splitting-off

Let G be a c-connected graph with m edges. By the sparse edge certificate construction of [20], in $O(m) \leq O(n^2)$ time we can find a subgraph G' with the same connectivity, but with total edge weight $O(nc)$. Thus $d(v) = O(nc)$ for all $v \in G'$, bounding the time for splitting-off by $\tilde{O}(n^3)$. The minimum of $d(v)$ over the vertices of G' is exactly c. In some applications, we may always split such a vertex off; then the time of splitting-off is just $\tilde{O}(n^2)$.

Our algorithm for the *graph orientation* problem takes advantage of this improved running time. Recall that we want to find a k-connected orientation of an undirected $2k$-connected graph. We use and sketch the steps of Mader's algorithm [18] (see also [17, problem 6.54]). We recursively orient graphs G' obtained from G, in one of the following two steps. If the graph has a degree $2k$ vertex s, we split s off. Otherwise we remove an edge with the property that the graph remains $2k$-connected. In both cases, it is simple to orient G, given the orientation of G'.

Now our algorithm is the following. We find the sparse edge certificate G' by the algorithm of [20]. Then G' has a vertex with $d(v) = 2k$, which can be split off in $\tilde{O}(n^2)$ time. Both steps are used $O(n)$ times; thus graph orientation can be done in $\tilde{O}(n^3)$ time.

Conclusions and open problems

We showed that approximately minimum cuts in an undirected graph can be described by a compact data structure. We also demonstrated with algorithmic applications that knowing approximately minimum cuts may speed up algorithms for connectivity-related problems. The Karger–Stein algorithm provides a sublinear-sized implicit representation for all approximate min-cuts; perhaps the most important practical application of our results is a query structure built within the same time for cuts found by that algorithm. Our result may also serve as a proof technique, especially related to splitting-off theorems and techniques.

Several theoretic and algorithmic questions related to near-minimum cuts remain open. Can our procedure yield improved s–t flow (or, more likely, Gomory–Hu tree [11]) algorithms? Or can we propose a new algorithm for performing the algorithmically very hard theorem of Mader (Theorem 1.3)? For a flow value upper bounded by v, it might be possible to find a min-cut of value v, by computing representations $O(\log v/c)$ times. The Karger–Stein algorithm for finding near-minimum cuts is in RNC. The question with the highest chance of a positive answer is whether one can build our representation in RNC as well.

A more theoretical open question is the enumeration of δ-near-mincuts. A variety of results [4, 26, 16, 13, 21, 25] address this question. The strongest known results are those of Karger [13] showing that the number of δ-near-mincuts is $O(n^{2\delta})$; of Nagamochi, Nishimara and Ibaraki [21] proving the tight bound $\binom{n}{2}$ for the number of 4/3-near-minimum cuts; and of Rauch and Williamson proving the tight asymptotics $O(n^2)$ for 3/2-near-mincuts. A cycle is an example showing that Karger's result is tight if δ is half-integer. However, tight result for general fractional values is not known. It also remains open whether Δ-near-minimum cuts for $\Delta > 6/5$ (e.g. $4/3$) have a well-characterizable structure.

Further, yet unexplored, features of approximate mincuts could be the following. In network reliability problems, edge weights of a graph may change independently at random; then mincuts become near-mincuts; see e.g. [14]. In such applications, our representation could be used instead of the cactus representation of all min-cuts [4]. Another example where the knowledge of a representation of near-mincuts may be useful is the cutting-plane algorithms for solving the traveling salesman problem [24, 3]. The planarity of our representation is perhaps also worth to be exploited, since in general, it is easier to handle planar than general graphs.

Acknowledgments

Many thanks to András Frank for stimulating discussions. Michel Goemans was particularly helpful in commenting various versions of the paper.

References

[1] Benczúr, A.A., Augmenting edge connectivity in $\tilde{O}(n^3)$ time, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994), 658–667; journal version in preparation.

[2] Benczúr, A.A., *The Structure of Near-Minimum Edge Cuts*, tech. rep. MIT/LCS/TR-639 (1994).

[3] Cook, W., lecture, *Summer School on Combinatorial Optimization*, Maastricht, August 1993.

[4] Dinits, E.A., A.V. Karzanov and M.L. Lomonosov, On the structure of a family of minimal weighted cuts in a graph, *Studies in Discrete Optimization* (in Russian), A.A. Fridman (Ed), Nauka, Moscow (1976), 290–306.

[5] Dinits, Y. and Z. Nutov, A 2-level cactus model for the system of minimum and minimum+1 edge-cuts in a graph and its incremental maintenance, *Proc. 27th Annual ACM Symp. on Theory of Comp.* (1995), 509–517.

[6] Edelsbrunner, Herbert, *Algorithms in combinatorial geometry.* Springer-Verlag (1987).

[7] Frank, A. On a Theorem of Mader, Proc. of the "Petersen" Graph Conference, Denmark (1990), J. Bang-Jensen, T. Jensen, L.K. Joergensen, B. Toft and P.D. Vestergaard, eds., *Annals of Discrete Mathematics* **101** (1992), 49–57.

[8] Frank, A., Augmenting graphs to meet edge connectivity requirements, *Proc. 31st Annual IEEE Symp. on Foundations of Comp. Sci.* (1990), and *SIAM J. Discr. Math.* **5** (1992) No. 1, 25–53.

[9] Frank, A., Applications of submodular functions, *Surveys in Combinatorics*, London Math. Soc. Lect. Notes, Ser. 187, K. Walker, Ed., Cambridge (1993), 85–136.

[10] Gabow, H.N., Efficient Splitting Off Algorithms for Graphs, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994), 696–705.

[11] Gomory, R.E. and T.C. Hu, Multi-terminal network flows, *SIAM J. Appl. Math.* **9** (1961), 551–560.

[12] Goemans, M.X. and V.S. Ramakrishnan, Minimizing Submodular Functions over Families of Sets, *Combinatorica* **15** (1995).

[13] Karger, D.R., Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm, *Proc. of 4th ACM–SIAM Symp. on Discrete Algorithms* (1993).

[14] Karger, D.R., Random sampling in cut, flow and network design problems, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994).

[15] Karger, D.R. and C. Stein, A new approach to the min-cut problem, *draft journal version.* Preliminary version: An $\tilde{O}(n^2)$ Algorithm for Minimum Cuts, *Proc. 25th Annual ACM Symp. on Theory of Comp.* (1993), 757–765.

[16] Lehel, J., F. Maffray and M. Preissmann, Graphs with maximal number of minimum cuts, *Technical report RR 880-M, Grenoble* (1992)

[17] Lovász, L., *Combinatorial Problems and Exercises*, 2nd edition, North-Holland, Amsterdam (1993).

[18] Mader, W., A reduction method for edge-connectivity in graphs, *Annales Discr. Math.* **3** (1978), 145–164.

[19] Mulmuley, K., Lower bounds for parallel linear programming and other problems, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994), 603–614.

[20] Nagamochi H. and T. Ibaraki, A linear time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph, *Algorithmica* **7** (1992), 583–596.

[21] Nagamochi H., K. Nishimara and T. Ibaraki, "Computing all small cuts in undirected networks", *Proceedings of the 5th ISAAC, Beijing, P.R. China, 1994*, Lecture Notes in Computer Science, **834**, D.-Z. Du and X.-S. Zhang, eds., Springer-Verlag, 190–198 (1994).

[22] Naor, D., D. Gusfield and Ch. Martel, A fast algorithm for optimally increasing the edge connectivity, *Proc. 31st Annual IEEE Symp. on Foundations of Comp. Sci.* (1990), 698–707.

[23] Naor, D. and V.V. Vazirani, Representing and enumerating edge connectivity cuts in RNC, *Proc. Second Workshop on Algorithms and Data Structures* (1991), Lecture Notes in Computer Science **519**, Springer–Verlag, pp. 273–285.

[24] Padberg, M. and G. Rinaldi, Facet identification for the symmetric traveling salesman polytope, *Math. Programming* **47** (1990), 85–136.

[25] Rauch Henzinger, M. and D.P. Williamson, *On the Number of Small Cuts in a Graph,* Technical Report No. 95–1486. Department of Computer Science, Cornell University (1995).

[26] Vazirani, V. and M. Yannakakis, Suboptimal cuts: Their enumeration, weight, and number, *19th International Colloquium on Automata, Languages and Programming* (1992), *Lecture Notes in Computer Science* **623**, Springer-Verlag, 366–377.

[27] Watanabe, T. and A. Nakamura, Edge connectivity augmentation problems, *Comp. System Sci.* **35** (1987), 96–144.